NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

REAL-TIME, REMOTELY CONTROLLED, UNMANNED, SURFACE COMBATANT (RT-RCUSC) USING THE INTERNET

by

Floyd Bailey
&
Carl Robbins

September, 1997

Thesis Advisor: Luqi

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE: REAL-TIME, REMOTELY CONTROLLED, UNMANNED, SURFACE COMBATANT (RT-RCUSC) USING THE INTERNET

5. FUNDING NUMBERS

6. AUTHOR(S) Floyd Bailey & Carl Robbins

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis was developed in response to the Navy's goal to reduce staffing levels aboard surface combatants. The thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant, (RT-RCUSC) possible using wire and wireless Internet connections and protocols.

A Command and Control (C2) model was developed for using the rapid prototype methodology. The C2 model collected latency data which was analyzed to determine the feasibility of a RT-RCUSC.

Sixteen experiments using latency times were designed to determine the viability of communication paths that progressively increased in distance and complexity. Variables included the use of two protocols, TCP and UDP, the use of two satellite types, geosynchronous and Low Earth Orbiting (LEO), as well as employing up to two satellites per end-to-end transmission path.

The results demonstrated that real-time control of a ship's navigation system can be performed when entries are made directly on the server PC or when using a client PC that is connected to the server PC via an Ethernet LAN. When controlling RT-RCUSC directly from the server using TCP and UDP the mean latency time was approximately 31.4 and 32.0 milliseconds respectively with the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly when controlling RT-RCUSC from a client, connected to the server via a LAN, using TCP and UDP, the mean latency time was approximately 32.0 and 31.1 milliseconds respectively with the greatest latency time equal to 90 and 100.5 milliseconds respectively. Security restrictions prevented Java socket formation between the client/server interface when testing wireless paths aboard the USS Coronado. The restrictions prevented us from gathering latency data for our geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite communications transmission/reception equipment failed to be provided for our evaluation. Therefore we were unable to gather latency data for our LEO wireless connection experiments.

Future research needs to focus on gathering latency data from both geosynchronous and LEO satellites for the purposes of determining the viability of a RT-RCUSC in order to determine the effectiveness of wireless communications with respect to Naval C2 systems.

14. SUBJECT TERMS Real-Time, Remote Control, Unmanned, Surface Combatant, Internet, Latency.

15. NUMBER OF PAGES 223

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

REAL-TIME, REMOTELY CONTROLLED, UNMANNED, SURFACE COMBATANT (RT-RCUSC) USING THE INTERNET

Floyd Bailey
B.S., University of Idaho, 1983
&
Carl Robbins
B.S., San Diego State University, 1978

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

September 1997

Authors: _____
Floyd Bailey

_____
Carl Robbins

Approved by: _____
Luqi, Thesis Advisor

_____
Valdis Berzins, Second Reader

_____
Ted Lewis, Chairman
Department of Computer Science

ABSTRACT

This thesis was developed in response to the Navy's goal to reduce staffing levels aboard surface combatants. The thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant, (RT-RCUSC) possible using wire and wireless Internet connections and protocols.

A Command and Control (C2) model was developed using the rapid prototype methodology. The C2 model collected latency data which was analyzed to determine the feasibility of a RT-RCUSC.

Sixteen experiments using latency times were designed to determine the viability of communication paths that progressively increased in distance and complexity. Variables included the use of two protocols, TCP and UDP, the use of two satellite types, geosynchronous and Low Earth Orbiting (LEO), as well as employing up to two satellites per end-to-end transmission path.

The results demonstrated that real-time control of a ship's navigation system can be performed when entries are made directly on the server PC or when using a client PC that is connected to the server PC via an Ethernet LAN. When controlling RT-RCUSC directly from the server using TCP and UDP the mean latency time was approximately 31.4 and 32.0 milliseconds respectively with the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly when controlling RT-RCUSC from a client, connected to the server via a LAN, using TCP and UDP, the mean latency time was approximately 32.0 and 31.1 milliseconds respectively with the greatest latency time equal to 90 and 100.5 milliseconds respectively. Security restrictions prevented Java socket formation between the client/server interface when testing wireless paths aboard the USS Coronado. The restrictions prevented us from gathering latency data for our geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite communications transmission/reception equipment failed to be provided for our

evaluation. Therefore we were unable to gather latency data for our LEO wireless connection experiments.

Future research needs to focus on gathering latency data from both geosynchronous and LEO satellites for the purposes of determining the viability of a RT-RCUSC in order to determine the effectiveness of wireless communications with respect to Naval C2 systems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

To Dr. Luqi, we would like to express our most profound thanks for your many hours of patient guidance. This is especially true because of the challenges presented to all of us when employing the distance learning method. We wish you the best in your future endeavors.

To the Naval Research and Development activity (NRaD), which is the Research, Development, Test and Evaluation (RDT&E) Division of the Naval Command Control and Ocean Surveillance Center (NCCOSC) for supporting this distance learning program with the Naval Post-Graduate School and proving the concept that investment in employees yields substantial returns to the employer.

To Debbie, Carl's wife, and to Carl's daughters, Valerie and Elizabeth, thank you all for your support and patience over the past two years. In the absence of such support this journey would not have been possible.

To our parents whose sacrifices allowed us to pursue our dreams.

I. INTRODUCTION

A. PROBLEM STATEMENT

The traditional unpopularity of U.S. military service personnel casualties combined with the low numbers of casualties recorded during Operation Desert Storm through the employment of U.S. developed high technology has resulted in an even greater demand on that technology to support the military. Furthermore, Operation Desert Storm proved that when an objective is quickly obtained there are significantly lower levels of U.S. military service personnel casualties. High tech "smart" weapons can rapidly suppress hostilities resulting in decreased casualties. This form of rapid and successful warfare has had a tremendous impact on the U.S. citizens. As a result not only do U.S. citizens insist on their approval prior to a conflict they also demand quick, decisive and successful action with minimal casualties. Therefore politicians and senior military personnel must respond to those demands in order to gain support before they commit U.S. forces.

With the collapse of the Soviet Union and the apparent end of the Soviet Communist Threat, demands have been placed on the U.S. military to reduce its size. This combined with the additional pressures that are being placed on our government to reduce spending in order to address the enormous Federal budget deficit have resulted in tremendous decreases in military and civilian personnel, equipment (ships, planes, etc.) and research.

The U.S. military is faced with a formidable challenge. With less money the U.S. military must resolve hostilities quickly, with a minimum of casualties, using fewer people and less equipment. In short, the U.S. military is being asked to do more with less.

The most obvious option is to employ the technology that is presently available. Coincidentally this is the objective of the current Smart Ship Project. The Smart Ship Project is an integrated product team established in November 1995 in the Naval Surface Warfare Center in Maryland for pursuing the following charter: "Develop, evaluate, and

select solutions to demonstrate that reductions in the crew's workload for a surface combatant can be achieved. Solutions will involve the application of available technology, changes to personnel and manpower policies, and changes to any other policies or procedures which drive shipboard manpower requirements. Implement these solutions using a pilot program on an operational ship and evaluate the ship's ability to maintain readiness and accomplish its mission. Identify specific billets which can ultimately be eliminated." [Ref. 1] [Fig. 1]

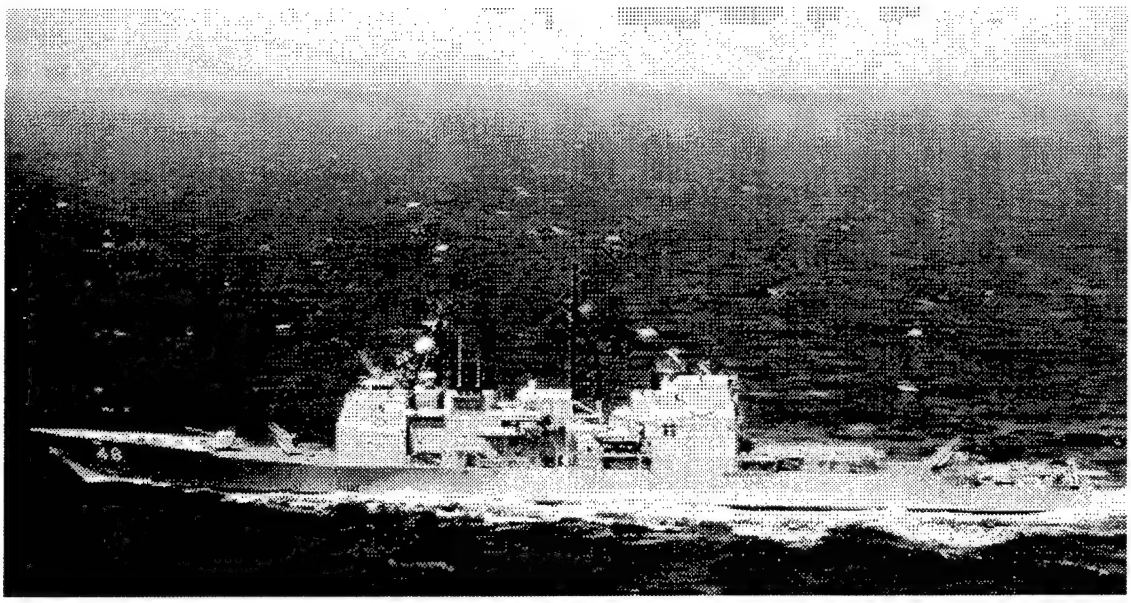


Figure 1. USS Yorktown, CG-48, the Smart Ship Test Platform
(Courtesy, Office of the Secretary of Defense Public Affairs)

This thesis takes the Smart Ship goal to the limit by exploring the ultimate solution in reducing the staffing requirements aboard a Navy ship. This thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant, RT-RCUSC (pronounced "RT-RUCKUS") possible using wire and wireless Internet connections and protocols. The communications network used to support this system is described.

We contend that it is not only possible but practical to develop a RT-RCUSC. By eliminating all personnel from a naval combatant the number of U.S. casualties will be

reduced even if the combatant is disabled or destroyed. This approach is the ultimate solution in safeguarding against shipboard casualties should the combatant be placed in harms way.

Therefore the purpose of this thesis is to present the concept of a Real-Time Remotely Controlled Unmanned Surface Combatant (RT-RCUSC). RT-RCUSC is our contribution to the challenges that face today's U.S. Navy by posing a possible solution to the demands listed above. Our RT-RCUSC is an alternative to the traditional approach of sending both men and ships in harms way. Of course the byproduct of substituting U.S. military personnel with technology is that we significantly reduce the opportunity for casualties within our forces.

B. RAPID PROTOTYPING

Prototyping of hardware has long been accepted in the engineering discipline yet it remains relatively unused in software development. By applying the rapid prototyping methodology to our RT-RCUSC we can develop a model that will allow us to evaluate the feasibility of the RT-RCUSC without having to specify the entire system or develop all of the code [Ref. 2-5]. Lacking the need for detailed design, the RT-RCUSC development and evaluation can be accomplished very quickly through the use of iterations of the model where each iteration can be tested, verified and analyzed.

C. METHODOLOGY & DELIVERABLES

The RT-RCUSC was developed by researching the existing technology and previous applications of unmanned, remotely controlled devices, defining an overall architecture, specifying a subsystem, rapidly prototyping the software, integrating the hardware and software elements, testing and reporting on the results.

The deliverables are an executable prototype, source code, and this thesis write-up which evaluates the effectiveness of the model. The model is available for review by any

of the DOD and C4ISR stakeholders, with the intention of performing subsequent refinements as well as the substitution of actual classified parameters.

D. THESIS ORGANIZATION

Chapter II provides sufficient background information to make this thesis a stand-alone document. Chapter III describes the design of the network architecture and the strategy behind the communications employed by the model. Chapter IV provides the details of the RT-RCUSC model including a report of the experiment. Finally, Chapter V provides the summary and conclusion of the research along with a discussion of follow on research.

II. BACKGROUND

A. TASKING

Reducing staffing levels aboard surface combatants, while at the same time maintaining a high level of effective readiness and responsiveness to hostile and/or threatening conditions, is the goal of present and future Navy projects. These projects include the Self Defense Test Ship (SDTS), Arsenal or Maritime Fire Support Demonstrator Ship, Surface Combatant-21 (SC-21) and CVX. In response to this goal we accepted the task to evaluate the effectiveness of the Internet in controlling surface combatants remotely. The use of the Internet, combined with the use of military and/or commercial communication satellites, provides a path that makes remote control possible. However the demand for real-time reaction and guaranteed delivery of operator initiated commands brings into the question the ability of the Internet, using wireless satellite communications, to control a surface combatant remotely. Real-time requirements are of paramount importance to a surface combatant in several areas such as track detection and reporting, threat evaluation, weapon's assignment and air control.

A real-time Command and Control (C2) system is required in order to demonstrate the ability of the Internet to remotely control an unmanned surface combatant. In addition the C2 system must gather latency data to determine the feasibility of this approach. Therefore a model C2 system will be developed using the rapid prototype methodology for the purposes of this evaluation. The latency data will be analyzed in order to determine the feasibility of using the Internet for real-time, remote control of a naval surface combatant.

It is important to review the current and future military systems that make use of wireless communications in order to appreciate the level of technology that either exists or is planned. A summary of wireless military systems is provided in the subsequent sections.

1. Today's Military

Today the U.S. military utilizes wireless communications to control Unmanned Aerial Vehicles (UAVs) and a surface ship. Below is a summary of UAVs and surface ship that are controlled remotely.

a. UAVs

Presently the Air Force is employing the Pentagon's most advanced unmanned spy plane, known as the Predator, which is designed to give commanders an immediate picture of troop and weapons movements on the ground. [Ref. 6][Fig. 2] The latest versions of the aircraft have radar-imaging systems enabling it to "see" through cloud cover. It is controlled by a pilot on the ground who guides the 27-foot-long plane to a target several hundred miles away, from where it can transmit images via satellite to ground commanders. It can stay on station over a target for as long as 24 hours at an altitude of 23,000 feet.

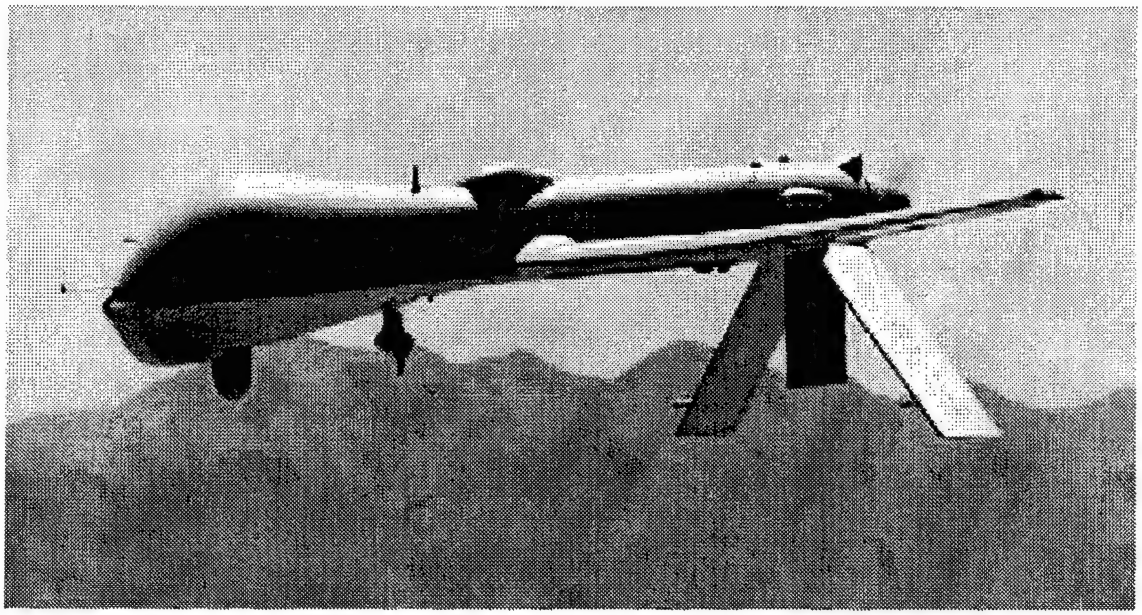


Figure 2. Predator UAV

(Courtesy, Office of the Secretary of Defense Public Affairs)

Additionally the Air Force is now developing another UAV. It is the Global Hawk high flying UAV, intended to provide military commanders with high-resolution, real-time imagery of large geographic areas. [Ref. 7][Fig. 3]

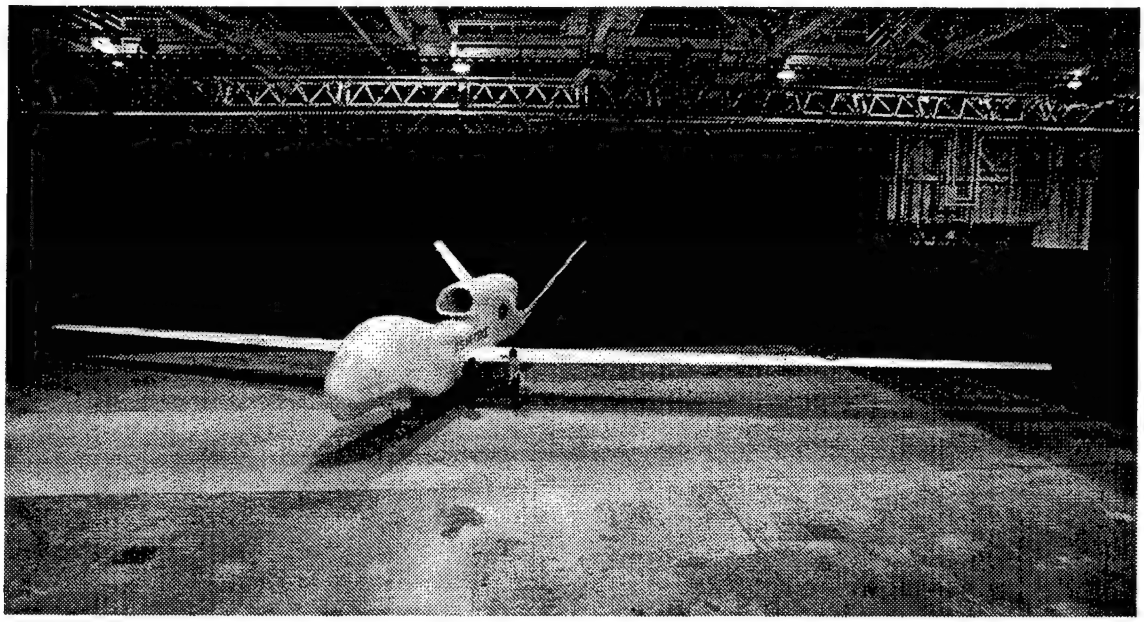


Figure 3. Global Hawk UAV

(Courtesy, Office of the Secretary of Defense Public Affairs)

Global Hawk is part of the Defense Advanced Research Projects Agency (DARPA) and Defense Airborne Reconnaissance Office's (DARO) High Altitude Endurance (HAE) UAV program, which is being pursued as an advanced concept technology demonstration.

The UAV is intended to operate in low-to-moderate threat environments in which it will be able to survey 40,000 square nautical miles in a single day from altitudes upwards of 65,000 feet.

According to the Pentagon, "For a typical mission, the Global Hawk system can fly to a target area 3,000 nautical miles away at 65,000 feet, and stay airborne for 24 hours collecting data before returning". Its long endurance capability permits the vehicle to view and track critical mobile targets for long periods. Potential alternative

payloads include signals-intelligence sensors, foliage-penetration radars and communications relay packages.

Global Hawk will be complemented by the Darkstar UAV being developed for the Air Force by Lockheed Martin and Boeing. [Ref. 7][Fig. 4] Darkstar will be optimized for high threat environments, as it contains low observable, or "stealth," characteristics.

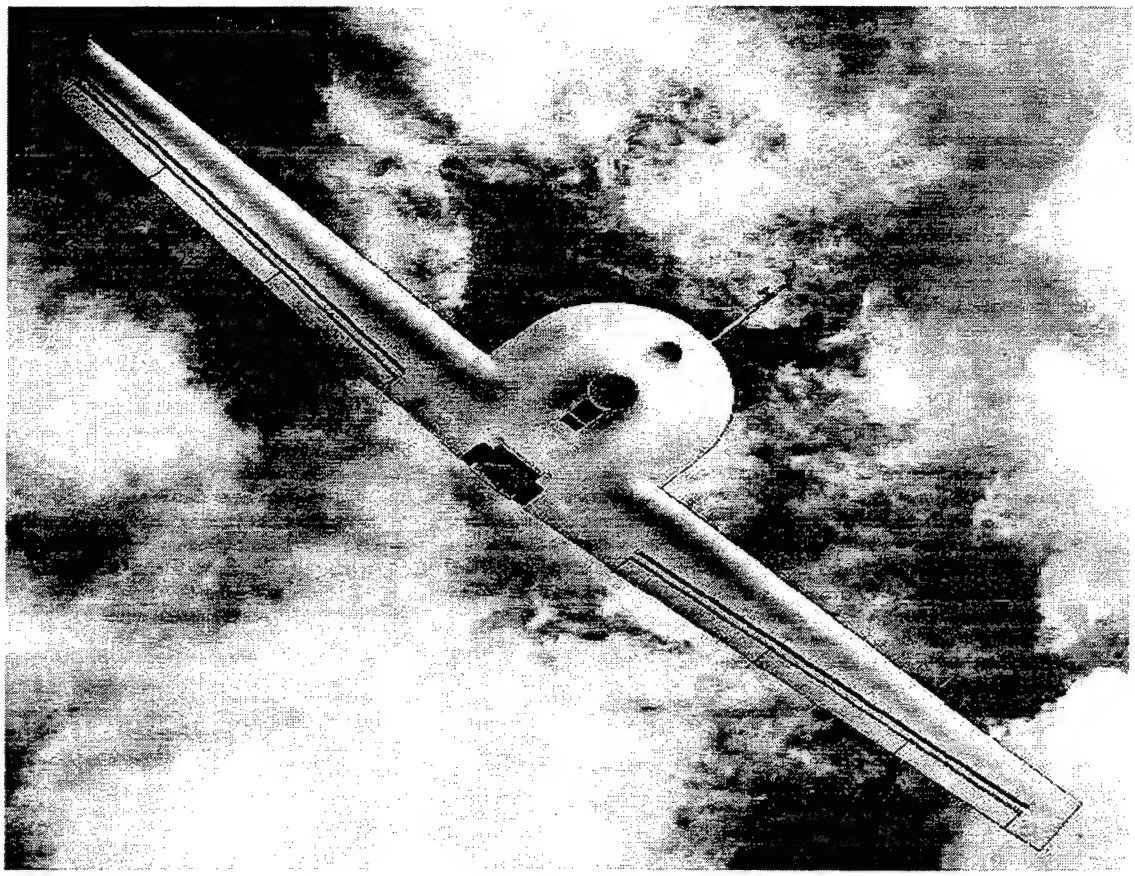


Figure 4. Dark Star UAV
(Courtesy, Lockheed Martin Corporation)

"Outrider" is the U.S. Army's UAV specialized to support Army intelligence gathering. Outrider is an advanced concept technology demonstration, aimed at getting a tactical spy drone fielded without going through the often lengthy conventional procurement process. [Ref. 8][Fig. 5]

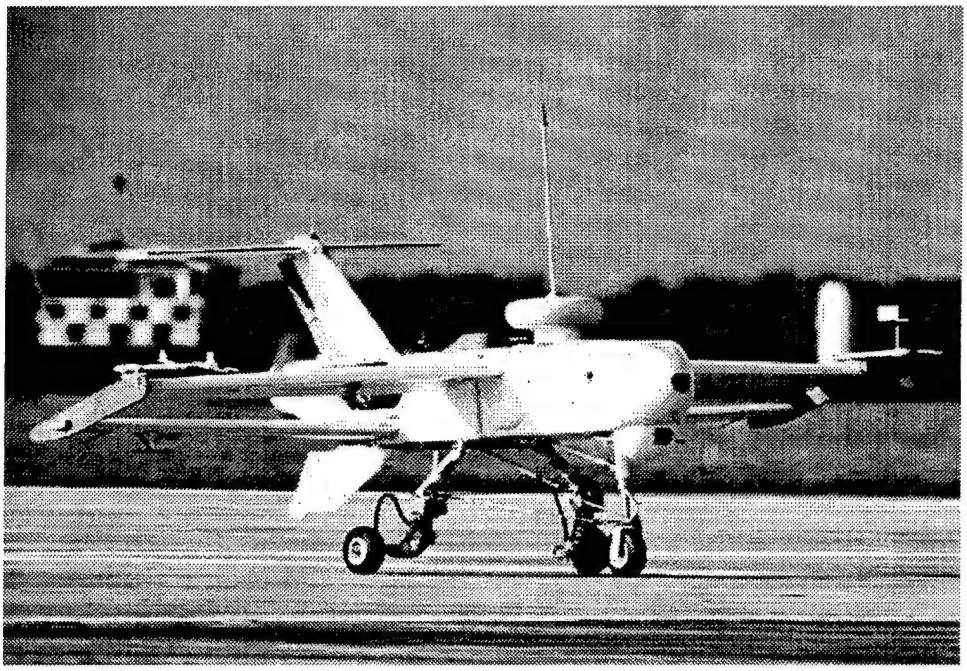


Figure 5. Outrider UAV

(Courtesy, Office of the Secretary of Defense Public Affairs)

The implication so far is that UAV's support only reconnaissance missions; however UAV's can also provide additional services.

Miniature UAV's are being developed that are the size of an adult's hand. [Ref. 9] These Micro Aerial Vehicles (MAV) can deliver ammunition, survey targets and inspect the inside of military buildings. The U.S. Defense Advanced Research Projects Agency (DARPA) is studying the feasibility of MAV's which measure about 15 cm (six inches) across. According to DARPA, "The sensors for these types of vehicle, if not here today, are within reach technologically and they represent a significant driver to want to build something small". MAVs are expected to be particularly useful in urban warfare where they could be employed to carry messages and carry out surveillance.

UAV's are being considered as a low cost cruise missile defense system. The UAV would be outfitted with sensors and a kill mechanism. [Ref. 10]

b. Naval Surface Combatants

Besides UAV's and MAV's, there is a naval surface ship that has been developed that can operate, unmanned, using remote control.

Originally decommissioned in 1983, the former USS DECATUR (DDG 31) was selected for conversion to a test ship in 1988. [Ref. 11][Fig. 6] Now known as the Self Defense Test Ship (EDDG-31), or SDTS, the ship is designed, primarily, for unmanned operation on the Pacific Missile Test Range. The SDTS can be piloted remotely, and its systems can be operated remotely, thereby eliminating the safety constraints which were required in previous testing. The ship is controlled remotely by the Weapons Division, Naval Air Warfare Center, Point Mugu, CA. The combat systems installed aboard SDTS are controlled remotely by the Port Hueneme Division, Naval Surface Warfare Center, Port Hueneme, CA.

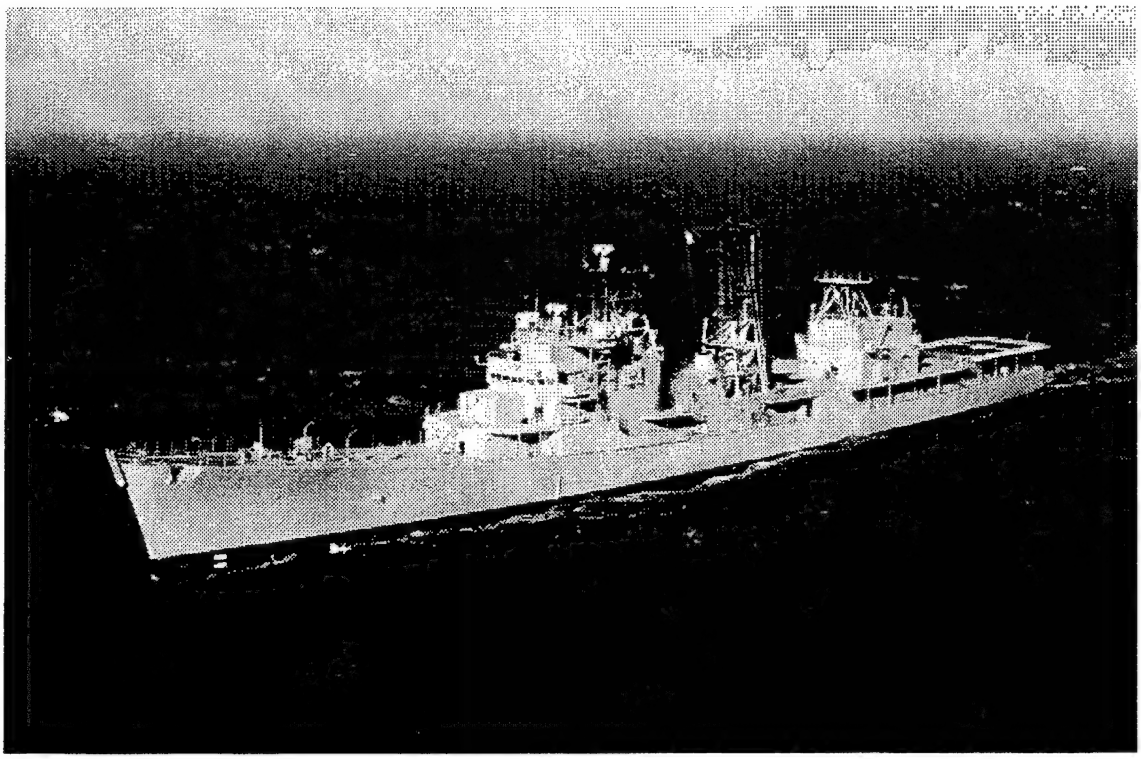


Figure 6. The former USS Decatur (DDG 31), now the Self Defense Test Ship (SDTS) (EDDG-31)

(Courtesy, U.S. Navy, via NavPhoto Archives)

A crew can also go aboard to pilot the ship and operate the installed systems and equipment whenever necessary to meet test objectives.

During typical operations, air launched threats and/or surface launched threats will attack the SDTS. The system under test, whether a new combat system configuration or an individual element (sensors, weapon systems, etc.) will respond to these threats to defend the ship.

Multiple weapons attacking the SDTS can be detected by multiple sensors and engaged by multiple defensive systems. With SDTS the T&E community has the opportunity to perform the kind of realistic, integrated, synergistic testing which cannot be accomplished aboard a manned vessel. The SDTS can also be utilized for testing when tied to the dock, and in cooperative test efforts with systems installed at the PHD NSWC Surface Warfare Engineering Facility (SWEF).

With the ship pier-side, SDTS is a floating laboratory available for use without the in port scheduling problems associated with a manned vessel. Interoperability issues can be addressed in a shipboard environment, system operational computer programs can be verified, and hardware related problems can be resolved.

Underway the vessel presents two distinct operating modes: manned and unmanned. Under either condition the vessel is available on a dedicated basis, and testing will not be preempted for routine shipboard operations, training evolutions or higher priority missions.

Remotely monitored and/or controlled systems include: Mk 23 Target Acquisition System, Mk 57 NATO SEASPARROW (Dual Directors), Combat Systems Remote Control System, Hulk Integrated Target System (Ship Remote Control), Two Wire Automatic Remote Sensing Evaluation System, Ordinance Magazines, AN/SLQ-32 ESM, Close In Weapons System (CIWS), and RAM System.

Additional examples of unmanned, remotely controlled devices can be found in the areas of space and ocean exploration. The point is the technology is well proven and universally accepted.

2. The Future Navy... now to the year 2,000

The Arsenal or the Maritime Fire Support Demonstrator Ship concept is an outgrowth of the Navy's shift in focus from the open ocean to the littoral. [Ref. 12][Fig. 7] It is fully consistent with "Forward from the Sea," and proposes an innovative means to provide more decisive, responsive, and varied naval support to the land battle. Through concentration of massive firepower, continuous availability and application of netted targeting and weapons assignment, the Arsenal Ship concept would supplement the programmed force of carriers and Tomahawk-capable combatants and submarines. The ship would be specifically tailored to meet the heavy support challenge in the opening days of conflict, without having to surge non-deployed surface ships and submarines from the United States.

The Arsenal Ship, along with other forward deployed naval and joint forces, will most likely be the key to successful introduction as well as early employment of ground forces. Initially operating under the control and umbrella of regularly deployed Aegis combatants, the Arsenal Ship concept envisions providing the Unified Commanders-in Chief (CinC) improved capability to halt or deter an invasion, and if necessary, help enable the build-up of coalition land-based air and ground forces to achieve favorable conflict resolution. With a current vision of no more than a six-ship force, the Arsenal Ships will be stationed continuously forward, always available for rapid movement upon receipt of even the most ambiguous or limiting strategic warning. Much like our maritime pre-positioning force, the Arsenal Ship proposal calls for the ships to remain on station in support of a Unified CinC for indefinite periods without dependence on host nation support or permission.

Proponents say that the Arsenal Ship can help win big wars faster with even fewer U.S. casualties than occurred in Desert Storm. The ships would provide the Navy the ability to participate, perhaps decisively, in the first hours of a war without having to wait days or weeks for the Marines, Air Force and Army to roll in from outside the theater. It is believed that the sooner and harder the Navy can strike an enemy the faster the war will be blunted or even won.

The Arsenal ship will have as many as 750 vertical launch tubes packed with Tomahawks and other smart missiles. During war, the Arsenal Ships would serve as floating missile magazines.

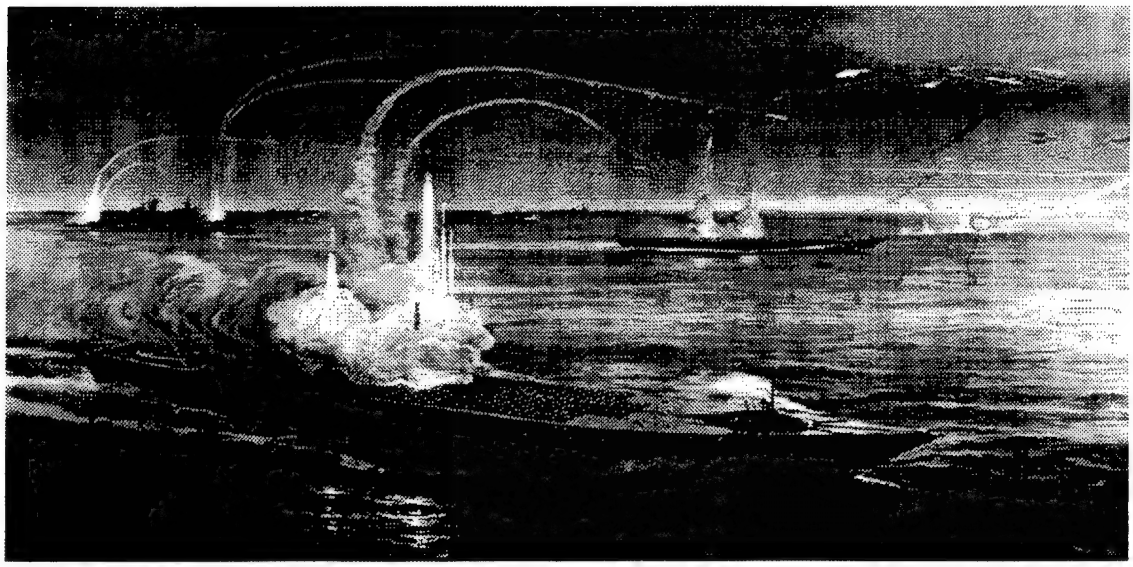


Figure 7. Arsenal or Maritime Fire Support Demonstrator Ship
(Courtesy, Office of the Secretary of Defense Public Affairs)

There will be no combat information center aboard the Arsenal Ship. All targeting, mission planning, command and decision functions will be made remotely from other ships, planes or ground stations. Air Force, Army or Navy controllers miles away, not the ship's crew, will target and fire its missiles. [Ref. 13] All of the Arsenal Ship's weapons should be able to be fired from remote locations, such as Arleigh Burke or Ticonderoga-class ships, aircraft carriers, or even Air Force planes and Army or Marine ground controllers. As few as 25 male and female sailors -- but no more than 100 -- will man the highly automated ship.

As envisioned, the Arsenal Ship's mission is to:

1. Deliver hundreds of smart missiles deep into enemy territory, smashing, slowing and stopping enemy tank and armored columns days before more Navy, Marine, Air Force and Army units can arrive from outside the combat theater.

2. Fire missiles able to intercept and destroy enemy ballistic missiles like the Scud and its more modern and destructive cousins.

3. Provide back-up air defense missiles to the fleet that will be launched by other ships and circling U.S. warplanes.

The Arsenal Ship defenses will be largely passive including a stealthy, radar evading design. Its survival will largely depend on destroyers and cruisers who will act as its defensive eyes and ears. Radar absorbing material and angular features will deflect enemy radar and be the ships' key design features. The ships might also be able to ballast down almost to the water-line to avoid visual or radar detection. Furthermore, if targeting can be done by other units using well-established downlink procedures, the ship need not even have much in the way of sensors, so stealth can be incorporated almost to the point of semi-submergence.[Ref. 14]

The emphasis on the Arsenal Ship is to act as a `bridge' to the SC 21 next generation `family of ships'.

3. The Future Navy... the year 2,000 and beyond

Future Navy warships, along with their combat, hull, mechanical and electrical systems, will be designed as a single integrated weapons system. [Ref. 15]

The "Surface Combatant 21" or SC21 approach attempts to accelerate the progress made in previous systems integration efforts (i.e. AEGIS) through what is called horizontal or total ship engineering (TSE). This revolutionary process regards the ship as a "system of systems" and provides built-in flexibility needed for insertion of future technologies.

SC21 is intended to be a family of ships which when combined will provide for the mission capabilities addressed in the Mission Need Statement (MNS)."

The SC21 MNS, approved by the Joint Requirements Oversight Council (JROC) in 1994, serves as the foundation for guiding 21st-century surface combatant design, research, development and acquisition program decisions, service and joint doctrine and cooperative efforts with U.S. allies.

The specific mission of SC-21, addressed in the MNS, is to carry the war to the enemy through offensive operations by:

1. Being able to launch and support precision strike weapons and to provide firepower support for amphibious and other ground forces.
2. Protecting friendly forces from enemy attack through the establishment and maintenance of battlespace dominance against theater missile, air, surface and subsurface threats.

SC21's multi-mission capabilities include: Power projection, Battlespace Dominance, Command, Control and Surveillance, Survivability, Mobility, Fleet Support Operations, Noncombatant Operations and Comprehensive Joint Requirements.

In accordance with the MNS, SC21 must employ a TSE architectural approach that optimizes life cycle costs and performance; minimizes operating conflicts; permits rapid equipment upgrades; allows computational and communication resources to keep pace with commercial technology; and provides the capability to fight even if damaged.

The TSE approach is expected to promote commonality among ship classes and make maximum use of open systems and modular design in the ship's infrastructure, while accounting for emerging technologies during the developmental phase. Significant reductions in personnel requirements also are expected to be realized through automation.

Finally, the MNS requires SC-21 meet specific operational constraints. Some of these include:

1. Being fully functional in all environments, such as heavy weather, in the presence of electromagnetic, nuclear, biological and chemical contamination, and/or shock effects from nuclear and conventional weapon attack.

2. Providing helicopter and unmanned aerial vehicle (UAV) landing and hangaring facilities and ammunition storage for operational support of multi-mission armed helicopters.

3. Integrating with other U.S. Navy, Marine Corps, joint and allied forces in combined, coordinated operations. Joint goals for standardization and interoperability with the widest number of weapon and sensor systems will be achieved to the maximum feasible extent.

4. Embarking Special Operations Forces when required for selected missions.

5. Transiting through the Panama Canal.

Capitalizing on recent advances in technology is of critical importance to SC-21 design efforts. Some specific areas/technologies SC21 expects to incorporate include: missile defense and cooperative engagement capabilities (TBMD/CEC); power projection capabilities; passive defensive (survivability) capabilities, such as stealth design or low radar cross section reduction, signal intercept exploitation and acoustic signature reduction; and command, control, communications, computers and intelligence (C4I) capabilities.

There will be an integrated satellite network, a data link network and a fire control network that will enable SC-21 combatants to communicate directly to the shooter ashore.

Although not specifically designed to be unmanned certainly requirements will be imposed to reduce crew size, thereby increasing the ship's dependence on technology. Although unlikely to be specified as a remotely controlled unmanned surface combatant it may be possible to specify this capability as a causality backup.

The CVX project also intends to bring present and future technology together in an effort to produce an Aircraft Carrier designed to meet the challenges of the future by relying heavily on emerging technology. [Ref. 16][Fig. 8]

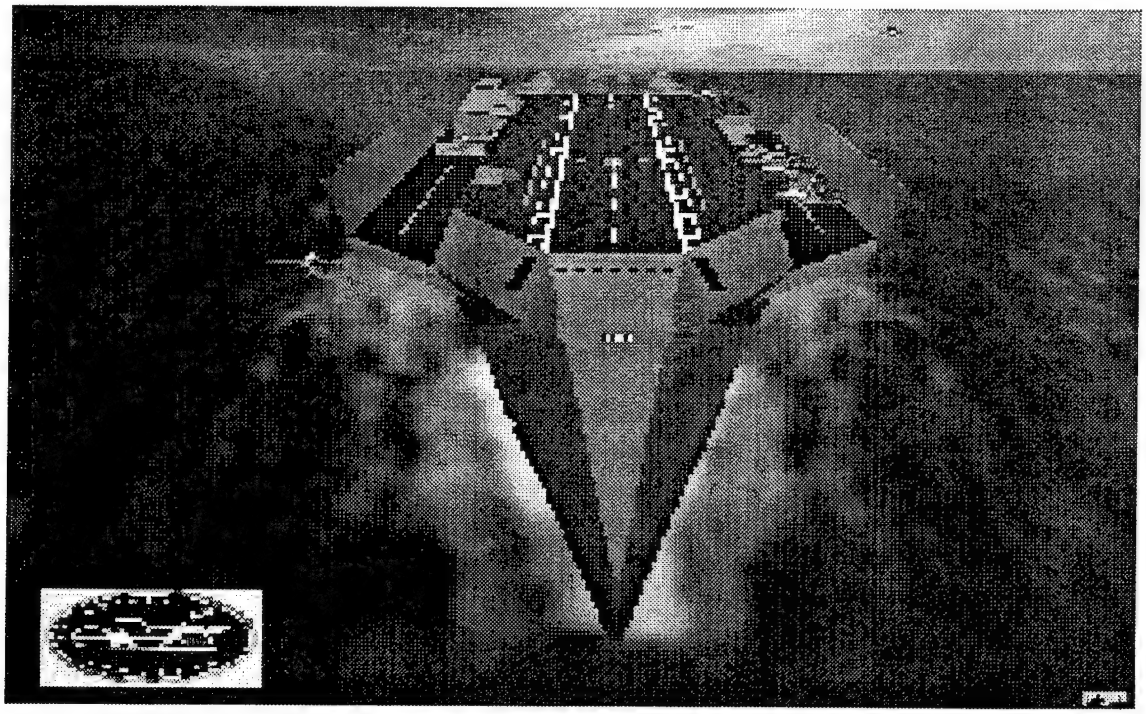


Figure 8. CVX

(Courtesy, Office of the Secretary of Defense Public Affairs)

4. More on Stealth...

Naval vessels always have had the distinct disadvantage of being quite visible and therefore vulnerable to attack and making all ships more stealthy, even aircraft carriers, is not new. [Ref. 17] Therefore making RT-RCUSC stealthy would become essential.

The United States has led the way since 1984 with its Lockheed-built Sea Shadow, a prototype patrol craft with its sides slanted upward at various angles to deflect radar. [Fig. 9] As was mentioned earlier the U.S. Arsenal, a concept ship that may one day serve as an unmanned, remote-controlled missile platform in the Persian Gulf that may also include a semi-surface profile.

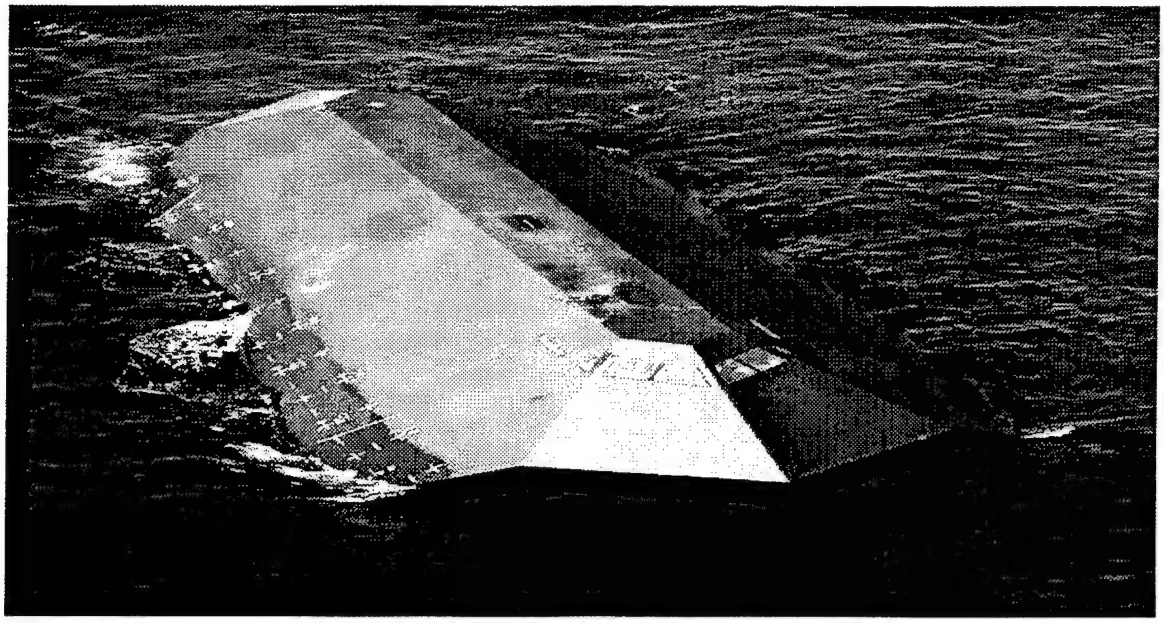


Figure 9. Sea Shadow

(Courtesy, Lockheed Martin Corporation)

France built stealth features into its new Lafayette class of frigates, whose hulls have diamond-like facets to thwart enemy radar and are coated with special radar-absorbent paint. Sweden has a similarly angular experimental patrol craft built of fiber-reinforced plastic.

Britian's Vosper Thornycroft's design of a 377-foot Sea Wraith stealth corvette, an anti-submarine patroller, whose multifaceted hull bears a striking resemblance to the stealth bomber, creates radar disturbances to throw oncoming missiles off-course, and its hull is designed to confuse enemy forces trying to track the vessel by radar and figure out what it is. [Fig. 10] On enemy screens, it's designed to have the "radar signature" of a small fishing boat.

Figure 10. Sea Wraith Stealth Corvette
(Courtesy, Vosper Thronycroft)

Its most intriguing feature is actually rather low-tech: nozzles that can generate a mist from the sea, hiding the ship from infrared search beams and masking hot spots that incoming heat-seeking missiles would try to sniff out.

Project Cougar, a model of a similar stealth ship design, was unveiled by BAeSEMA, a subsidiary of British Aerospace. The 311-foot steel vessel would target customers in the Pacific Rim region, where Singapore, Indonesia, Taiwan and others are building their fleets, the company said.

Designed for high-speed engagement in rough seas, that ship is propelled by jets of water to keep it quiet. Among its high-tech equipment: decoy launchers designed to seduce incoming enemy missiles away from the ship.

5. Impact of Information Technology for the 21st Century (IT-21) on the RT-RCUSC Design

Recently CINCPACFLT and CINCLANTFLT released a joint message concerning the development and implementation of IT-21. [Appendix A] To date they have provided IT-21 hardware/software implementation standards for programs that will be installing information systems on Fleet Units/Bases.

IT-21 is intended to support information superiority which has been determined to be the foundation of Joint vision 2010 battlefield dominance, as well as the war fighting vision for each service. IT-21 is a Fleet driven re-prioritization of C4I programs of record to accelerate the transition to a PC based tactical support war fighting network. The inactivation of the current DOD messaging system (AUTODIN) by Dec 99, with no planned navy infrastructure replacement, mandate the rapid implementation of this war fighting network.

The DOD Joint Technical Architecture (JTA) and Defense Information Infrastructure Common Operating Environment (DII COE) provide DOD with the Automated Information Systems (AISs) guidance required to take the navy into the 21st century. This convergence of solutions, problems and guidance provides the impetus to establish minimum Navy AIS standards at this time.

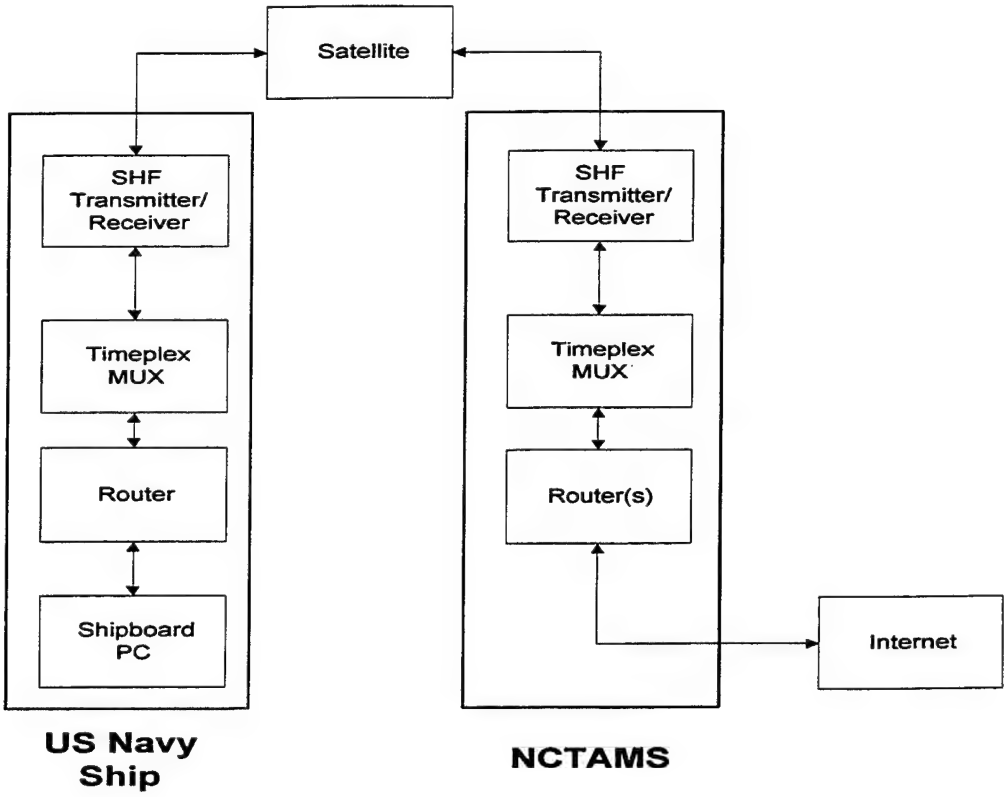
The IT-21 naval message defines all of the PC hardware and software that has been deemed acceptable. The list includes Windows NT, MS Exchange, MS Office 97, 32 bit operating systems, high resolution displays, mass storage, ATM backbone LANs with at least 100 mb/s etc. The IT-21 standards represent front end market technology, are dynamic in nature, and will continue to be closely linked to Commercial tends. The standards are intended to be minimum standards and will be updated periodically.

Therefore RT- RCUSC must assure compliance with the IT-21 message in order to gain Fleet acceptance.

6. Internet Connections Aboard Surface Combatants Today

Nearly every U.S. Navy ship today has connectivity to the Internet which provides E-Mail services and Internet access to shipboard personnel as well as offering ship unique Home Pages to land based "surfers" who may query a particular ship.

A shipboard user can access the Internet through a PC which is connected, via a router to a Timeplex Multiplexer (MUX) for message decomposition and scheduling which then passes the signal to the ship's satellite ultra-high frequency (UHF) transmitter/receiver equipment. The signal is then received by a satellite which forwards the signal to a land based UHF transmitter/receiver. The UHF transmitter/receiver is located at one of many ground stations. In this case the signal is received at a Naval Communications and Telecommunications Area Management Station (NCTAMS) and is recomposed through the NCTAMS's Timeplex MUX. From this point the signal is passed to one or more routers and is then allowed to pass onto the Internet. [Fig. 11]



NCTAMS = Naval Communications and Telecommunications Area Management Station

Figure 11. Current Shipboard Internet Architecture

7. Internet Connections Aboard Surface Combatants in the Future

A project by Bill Gates, chairman of Microsoft Corp., and Craig McCaw, a pioneer in cellular telephones, was awarded to Boeing Co. to coordinate the building of the Teledesic project. [Ref. 18] Teledesic would be used to build an "Internet in the sky" that would use hundreds of low-orbiting satellites to transmit data and conversations all over the world.

The plans call for Teledesic to begin high-speed two-way service-in which video and graphics would appear instantly on the computer screen-in the year 2002.

It is intended to bring closer an era of "personal" satellite communications, in which people can travel anywhere-even into the most remote regions of the world-and use a satellite telephone or link a computer to the Internet with no wires attached.

Such a system is intended to support the data rates and data volumes (bandwidth) necessary to allow for real-time, remote control, of an unmanned surface combatant.

ORBCOMM has recently deployed two Low Earth Orbiting satellites, the first in their constellation, which will provide capabilities similar to the proposed system underdevelopment by the Teledesic project. ORBCOMM has provided us two sets of equipment that will allow us to evaluate latency of LEOs.

At a recent symposium held at Stanford University entitled, "The Acceleration of World Wide Wireless Communications" companies such as Globalstar, ICO Global Communications, Bellcore, Telesis Technologies Laboratory, Netro, Hybrid, Hughes and NEC represent large companies that are all investing in the growth and improved services that can be obtained from wireless networks that support high speed integrated services in local, metropolitan and wide area environments. [Ref. 19]

B. JUSTIFICATION

Remote control of a surface combatant has been demonstrated using the SDTS, however its control is limited to the area contained within the Pacific Missile Range, specifically within the area covered by the line-of-sight RF transmitter towers located at San Nicholas Island, Point Hueneme and from Point Conception.

Internet connectivity offers over-the-horizon communications to naval surface combatants through satellite ultra-high frequency (UHF) links.

Therefore our stated hypothesis is that, "Real time, remote control, of an unmanned surface combatant is possible using the Internet when latency times are small enough to support real time control". In order to test our hypothesis our research will employ the use of a model that we will build, which will collect latency times from both geosynchronous and Low Earth Orbiting (LEO) satellites. The data will be gathered, analyzed, and used for comparing the performance of the two types of satellites. Finally

we will interpret the results and determine if real-time, remote control, of an unmanned surface combatant is feasible using the Internet. Incidentally in addition to the requirement of small latency times in order to support the real time remote control of an unmanned surface combatant we also realize the necessity for the requirement of sufficient bandwidth. However our efforts are limited to the latency aspect.

Referring to the Acceleration of World Wide Wireless Communications symposium it is obvious that improvements in world wide communications networks are of critical importance to a large vendor contingent. Therefore it is very likely that we do not understand the full impact of our research as the investment in world wide wireless communications is so substantial, and the amount of scientific data so scarce, that companies making these investments are likely to view this data closely.

Given these sets of circumstances we will attempt to produce a model that tests our hypothesis.

C. GOALS

The goal of the research is to perform an analysis of the real-time remote control of a surface combatant using the Internet and provide the following specifics:

1. A simplified model of the system.
2. A description of the system goals hierarchy and the functions it must perform.
3. Performance constraints on the system
4. Implementation constraints on the system.
5. Resource constraints for the development project.
6. The specification of the external interfaces of the major components.

D. REAL-TIME SYSTEMS

Real time combat system processes are those that require an event driven, deterministic response in action and reaction time, regardless of the system's state

(loading, process state, process time, and number of processes). [Ref. 20] Defined priority level, interrupt driven computer program architectures are usually required to provide event driven, deterministic responses vice multi-process computer program environments, which rely on cyclic status checking to determine if an event has occurred that requires a reaction. These architectures rely on machine speed to keep the cycle within reaction time requirements. In general, they are dependent on loading, process state, process time, and number of processes.

RT-RCUSC will require event driven, deterministic response in action and reaction time regardless of system state (loading, process state, process time, and number of processes).

Reasons for the real time requirement are guaranteed reaction time and known response. Related factors are state of the system at event time (loading, processes, etc.), data/action sequencing and need for graceful degradation.

Examples of typical events are radar contact detection, ID threat determination and engagement decision.

Timeliness issues are critical in real time systems. In particular two specific time intervals are of interest, service time and latency. [Ref. 21]

Service time is the net time taken to compute a response to a given event and is primarily a function of the algorithm used in the computation and is often deterministic and predictable.

Latency is the interval between the time of occurrence of an input and the time at which it starts being serviced. It takes into account a combination of different delays and is generally unpredictable.

Our model implements a small navigation algorithm with the specific goal of minimizing service time thereby allowing us to test 5 different configurations that RT-RCUSC may operate under so that latency data can more easily be obtained. Once the latency data has been obtained, and service time has been subtracted, we will perform an analysis on the latency data in order to determine the feasibility of RT-RCUSC.

In the construction of a deployable system both the service time and latency times for a given input are combined which yield the overall reaction time for that input. In

order to build effective real-time systems, especially those systems that have hard real-time requirements such as weapons system engagements and air control, it is important that the interval times always meets their prescribed deadline times. A hard real-time system requires that all deadlines must be met, otherwise the system is considered unacceptable.

E. SPECIFIC METHODOLOGY

Five specific experiments are to be conducted which will measure latency time between command initiation (client request) and command acknowledgment (server receipt). In accordance with the requirements set forth by the IT-21 Standards, PCs, that use Pentium processors, will be used. Additional equipment installed in the PCs will also meet IT-21 standards. The PCs will run under Windows NT, one a server, the other a client. The RT-RCUSC's home page, written in HTML and Java, as well as the interfacing pages will be installed on the server PC and will be accessed by the client PC. The server PC, whether simulated or actual, is intended to be installed aboard the RT-RCUSC. Communications between the client and server will make use of existing TCP/IP or UDP protocols and are accessed by the Internet Explorer web browser software package. Software will be included which will capture the latency times. The 5 experimental conditions are:

1. Simulated navigation control where command initiation is entered directly at RT-RCUSC's server. This experiment is intended to simulate control of the RT-RCUSC as if were being performed aboard the RT-RCUSC, directly at the RT-RCUSC's server PC. This is called the "server or direct control" experiment. [Fig. 12]

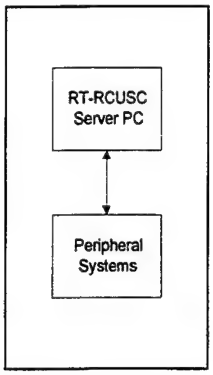


Figure 12. Server or Direct Control

2. Simulated navigation control where command initiation is entered into RT-RCUSC's client PC which, through communications via the shipboard LAN, accesses the RT-RCUSC server PC. This experiment is intended to simulate control of the RT-RCUSC as if were being performed aboard the RT-RCUSC from a client PC that sends commands to the server PC. This is called the "Shipboard Client Control" experiment. [Fig. 13]

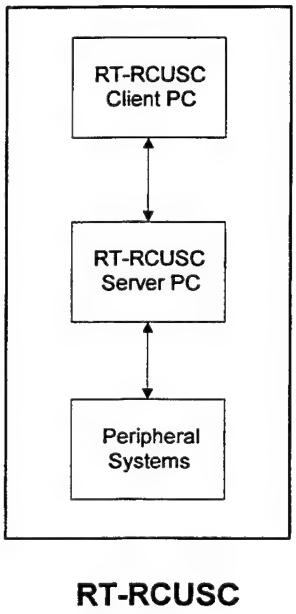
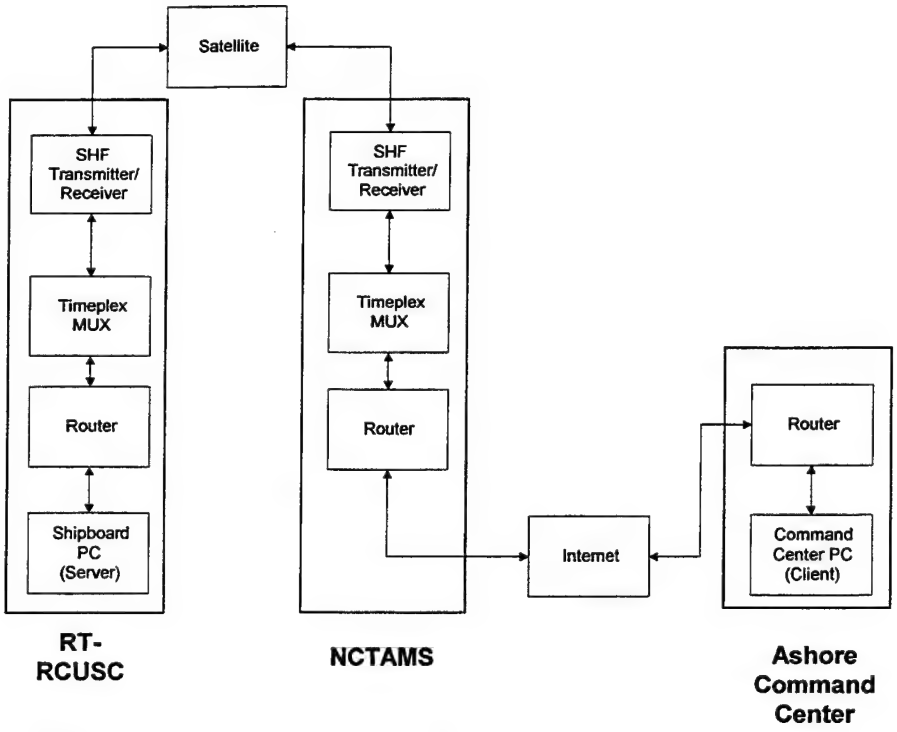


Figure 13. Shipboard Client Control

3. Simulated navigation control where command initiation is entered into a land based command center's client PC which passes the commands through the following path: the client's router, the Internet, the NCTAMS's router, the NCTAMS's Timeplex MUX, the NCTAMS's satellite UHF equipment, the satellite, the RT-RCUSC's satellite UHF equipment, the RT-RCUSC's Timeplex MUX, and ultimately to the RT-RCUSC's server which controls the peripheral equipment. This is called the "Ashore Command Center Control" experiment. It is intended to test the use of a single satellite where one (two way) wireless connection is employed. [Fig. 14]

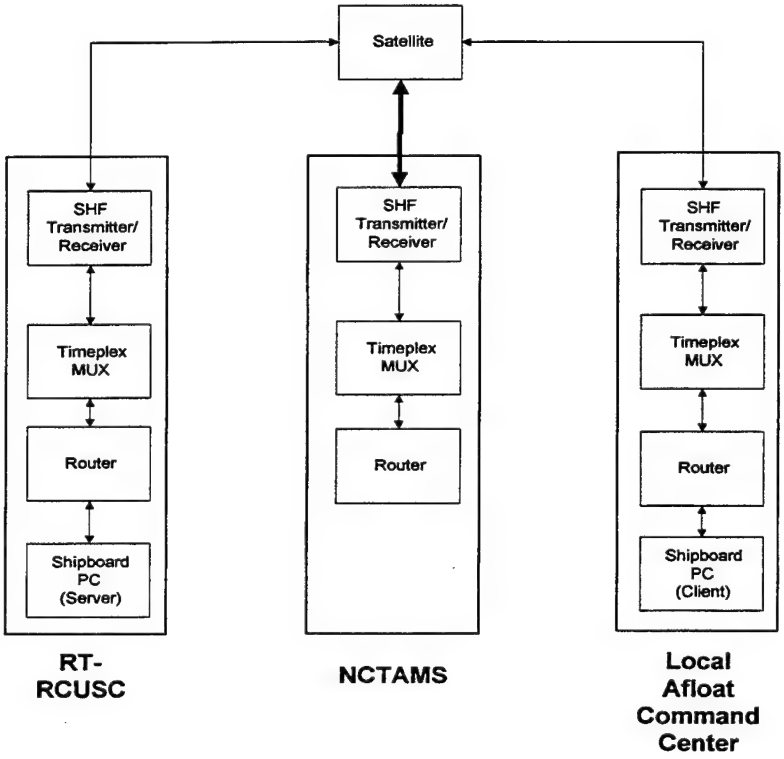


NCTAMS = Naval Communications and Telecommunications Area Management Station

Figure 14. Ashore Command Center Control

4. Simulated navigation control where command initiation is entered into a command ship's client PC which passes the commands through the following path: the

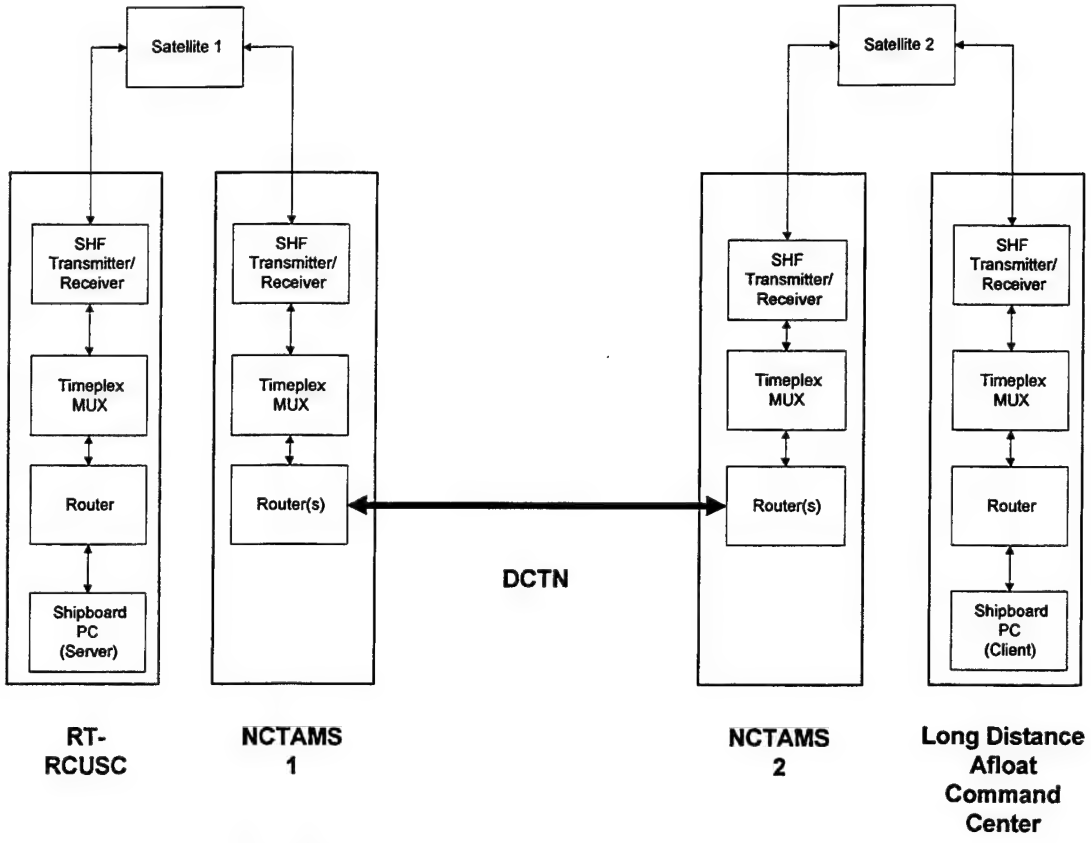
command ship's router, the command ship's Timeplex MUX, the command ship's satellite UHF equipment, the satellite, the NCTAMS's satellite UHF equipment, the NCTAMS's Timeplex MUX, the NCTAMS's router, the NCTAMS's Timeplex MUX, the NCTAMS's satellite UHF equipment, the satellite, the RT-RCUSC's satellite SHF equipment, the RT-RCUSC's Timeplex MUX, and ultimately to the RT-RCUSC's server which controls the peripheral equipment. This is called the "Local Afloat Command Center Control" experiment. "Local" refers to the condition where the same satellite and same NCTAMS are used by both the Local Afloat Command Center and by the RT-RCUSC when they are in communication with each other. It is intended to test the use of a single satellite where two (two way) wireless connections are employed. It would demonstrate the communications path that would be used within a battle group. [Fig. 15]



NCTAMS = Naval Communications and Telecommunications Area Management Station

Figure 15. Local Afloat Command Center Control

5. Simulated navigation control where command initiation is entered into a command ship's client PC which passes the commands through the following path: the command ship's router, the command ship's Timeplex MUX, the command ship's satellite UHF equipment, the satellite, a NCTAMS #1's satellite UHF equipment, NCTAMS #1's Timeplex MUX, NCTAMS #1's router, the Defense Communications and Telecommunications Network (DCTN), NCTAMS #2's router, NCTAMS #2's Timeplex MUX, NCTAMS #2's satellite UHF equipment, the satellite, the RT-RCUSC's satellite SHF equipment, the RT-RCUSC's Timeplex MUX, and ultimately to the RT-RCUSC's server which controls the peripheral equipment. The Defense Communications and Telecommunications Network (DCTN) provides a large and rapid data transfer capability through the use of dedicated commercial leased connections. This is called the "Remote Afloat Command Center Control" experiment. "Remote" refers to the condition where different satellites and different NCTAMS are used by the Remote Afloat Command Center and by the RT-RCUSC when they are in communication with each other. It is intended to test the use of a two satellites where two (two way) wireless connections are employed. It would demonstrate the communications path used for world wide control. [Fig. 16]



NCTAMS = Naval Communications and Telecommunications Area Management Station

DCTN = Defense Communications and Telecommunications Network

Figure 16. Long Distance Afloat Command Center Control

Timing data will be collected for each of the experimental conditions and the results will be analyzed in an effort to test the hypothesis.

III. NETWORK TOPOLOGY

A. NETWORK ARCHITECTURE FOR THE MODEL

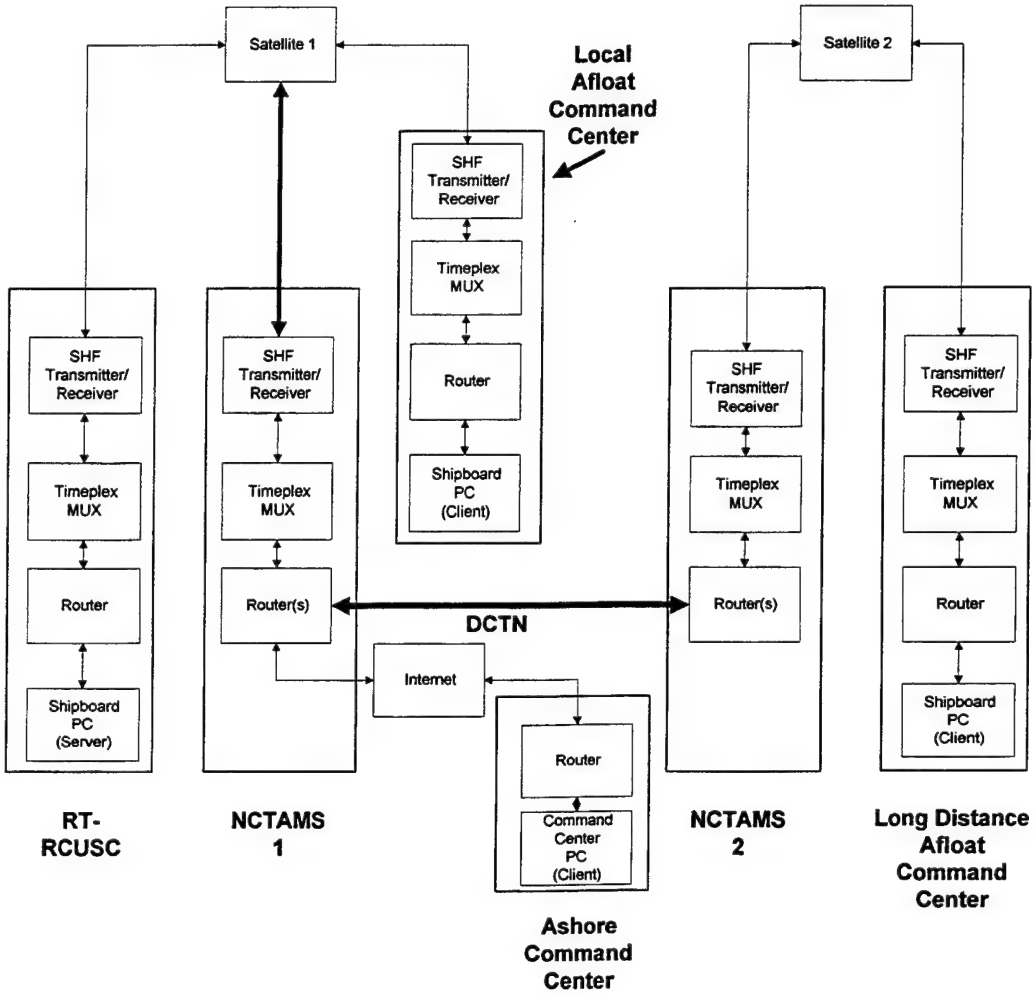
Installation of a PC which performs as a server running under Windows NT, a ship unique home page can be established which provides access, through a menu of hypertext links to other web pages that would offer a user the ability to control and monitor the various systems aboard the ship. In our model we will display the RT-RCUSC home page and offer only the Navigation Control hypertext link as a demonstrable entity. Other example systems will simply respond by saying, "This Site Under Construction".

An actual fielded version of a RT-RCUSC would require network security. Presently the use of SIPERNET by the Navy provides an adequate level of protection through the use of KG-194 encryption equipment. Although not available to us for our evaluation, future RT-RCUSC designs should include KG-194s in the construction of the network and timing delays included in the latency figures.

B. INTERNET COMMUNICATIONS STRATEGY

Figure 17 provides the overall diagram that details the specific requirements for the components of the model. The model support communications to RT-RCUSC either directly on the RT-RCUSC server, from a client that is aboard RT-RCUSC connected to the server via a LAN (not shown), from a Land Based Command Center, from a Local Afloat Command Center or from a Long Distance Afloat Command Center. Zero to two satellites are required in order to send commands to RT-RCUSC. The Land Based Command Center, the Local Afloat Command Center, the Long Distance Command Centers all require the use of the Naval Communications and Telecommunications Area Management Stations (NCTAMS) for access to the military communications satellites. In addition the Long Distance Afloat Command Center requires the use of the Defense

Communications and Telecommunications Network (DCTN) for transmission of information between NCTAMSs.



NCTAMS = Naval Communications and Telecommunications Area Management Station

DCTN = Defense Communications and Telecommunications Network

Figure 17. Overall Diagram Detailing the Physical Components of the Model

IV. RT-RCUSC MODEL

A. THE MODEL FOR ALL DEVICES AND COMMUNICATIONS LINKS

The RT-RCUSC model is a C2 application program which makes use of the Internet for command message routing. The Java programming language was selected as the language for implementing the RT-RCUSC model as it supports the concepts as outlined by the IT-21 standards and because it supports the development of Internet software. TCP and UDP were selected as the protocols for evaluation as they are the defacto standards for two way communications between Internet processes.

1. General Description

RT-RCUSC is made up of three major components: the supporting HTML web pages, sever and client software. The HTML web pages are created using HTML files. The server and client software are contained in Java files.

2. HTML Files

There are six HTML files for the TCP and UDP versions of RT-RCUSC. Main.html file is the Home Page for RT-RCUSC which provides the user the option of viewing a summary of the thesis or executing the RT-RCUSC model. [Appendix B, Fig. 18] Ths_Sum.html file is the web page displaying the thesis summary. [Appendix C, Fig. 19] The Sm4.html web page offers the user the option of selecting one of three C2 pages: Navigation, Weapons, or RADAR. [Appendix D, Fig. 20] The Nav_Mod.html web page downloads the applet that allows the user to interface with the RT-RCUSC model. [Appendix E, Fig. 21] Web pages Rad_Mod.html and Wea_Mod.html are place holder sites to be used in the future for the implementation of the radar and weapons control pages and are not currently implemented in our model. [Appendices F and G, Figs. 22 and 23]

RT-RCUSC (UDP)

The Real-Time, Remotely Controlled, Unmanned, Surface Combatant (**RT-RCUSC**) is the exploration of using Internet connections and protocols for control and communication between an unmanned ship and various controlling sites. See thesis description for more information. Thesis demonstration brings up a WEB site of options that are prototypes developed in this thesis used to collect timing information.

Figure 18. RT-RCUSC Home Page

RT-RCUSC Thesis Summary

This thesis was developed in response to the Navy's goal to reduce staffing levels aboard surface combatants. The thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant, (RT-RCUSC) possible using wire and wireless Internet connections and protocols.

A Command and Control (C2) model was developed using the rapid prototype methodology. The C2 model collected latency data which was analyzed to determine the feasibility of a RT-RCUSC.

Sixteen experiments using latency times were designed to determine the viability of communication paths that progressively increased in distance and complexity. Variables included the use of two protocols, TCP and UDP, the use of two satellite types, geosynchronous and Low Earth Orbiting (LEO), as well as employing up to two satellites per end-to-end transmission path.

The results demonstrated that real-time control of a ship's navigation system can be performed when entries are made directly on the server PC or when using a client PC that is connected to the server PC via an Ethernet LAN. When controlling RT-RCUSC directly from the server using TCP and UDP the mean latency time was approximately 31.4 and 32.0 milliseconds respectively with the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly when controlling RT-RCUSC from a client, connected to the server via a LAN, using TCP and UDP, the mean latency time was approximately 32.0 and 31.1 milliseconds respectively with the greatest latency time equal to 90 and 100.5 milliseconds respectively. Security restrictions prevented Java socket formation between the client/server interface when testing wireless paths aboard the USS Coronado. The restrictions prevented us from gathering latency data for our geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite communications transmission/reception equipment failed to be provided for our evaluation. Therefore we were unable to gather latency data for our LEO wireless connection experiments.

Future research needs to focus on gathering latency data from both geosynchronous and LEO satellites for the purposes of determining the viability of a RT-RCUSC in order to determine the effectiveness of wireless communications with respect to Naval C2 systems.

Figure 19. Thesis Summary Page

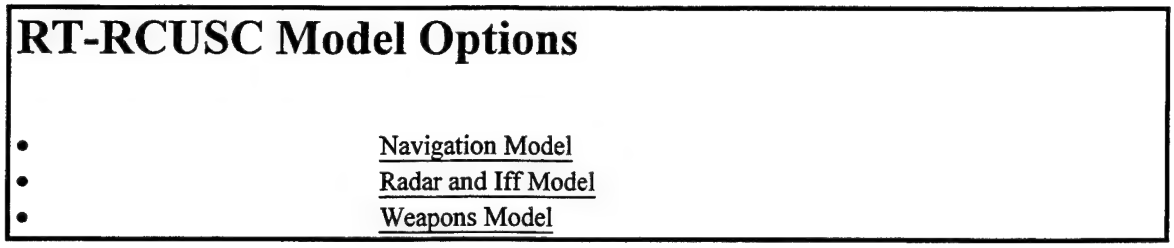


Figure 20. SM4.html

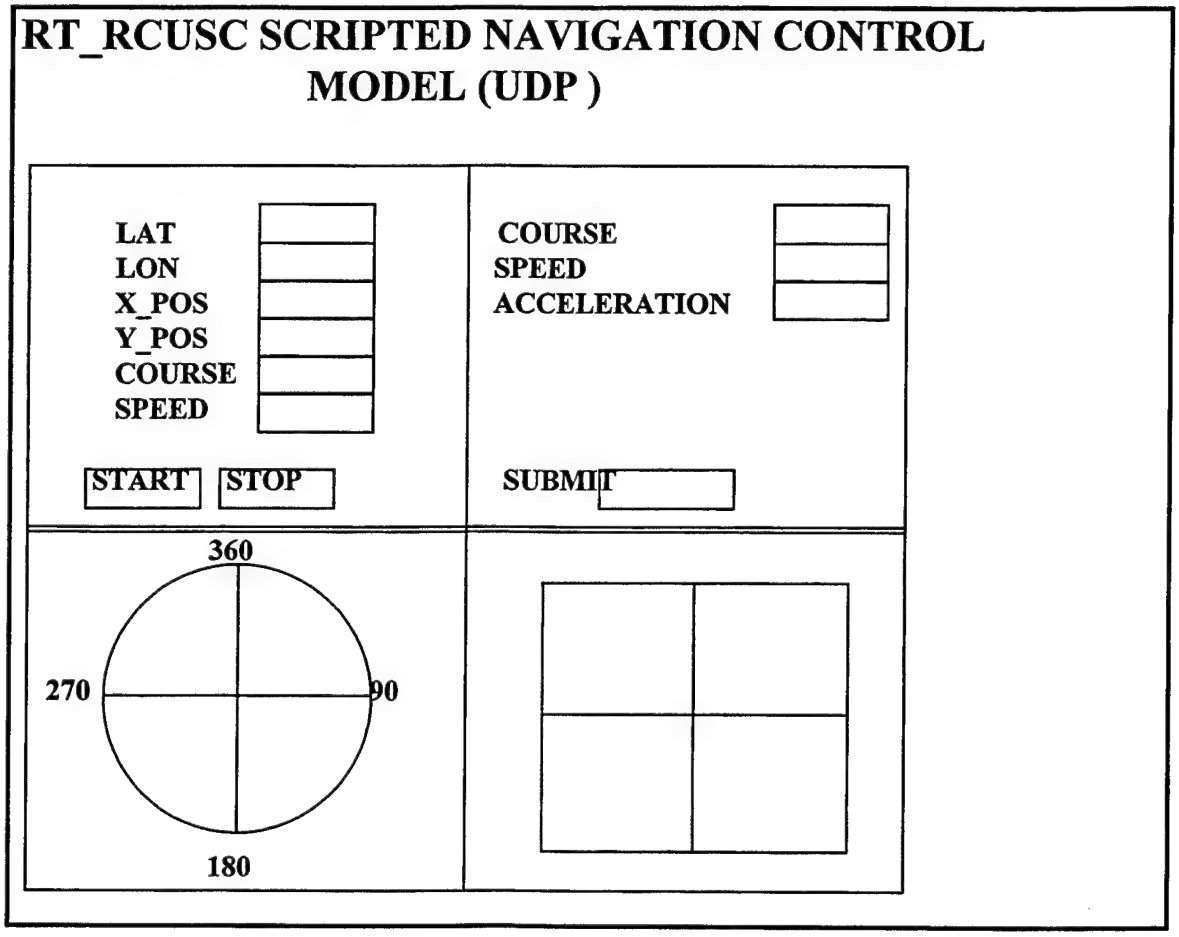


Figure 21. Nav_Mod.html

Figure 22. Rad_Mod.html

RT-RCUSC WEAPONS MODEL

UNDER CONSTRUCTION

Figure 23. Wea_Mod.html

3. Java Files

In addition to the HTML files, RT-RCUSC requires three Java files: RT_RCUSCHpage.java, RT_Server.java and Smart_Constants.java.

a. Java Classes for RT_RCUSCHpage.java

RT_RCUSCHpage.java is the client portion of the RT-RCUSC model and is made up seven Java classes. [Appendix I for TCP, Appendix L for UDP] The class RT_RCUSCHpage is the driver for the client. This class is responsible for instantiation of the NavControl, Net_Connection, Nav_Message, NavUpdate, Compass and Plot classes. [Fig. 24] RT_RCUSCHpage provides periodic and non periodic commands, from the client to the server, such as "START", "STOP", course, speed and positional changes.

The NavControl and NavUpdate classes accept and validate user entered data. Nav Control initiates and terminates the simulation program contain within RT_Server. NavUpdate processes user entered changes once the system has begun execution.

The Compass and Plot classes display the heading and position of the RT-RCUSC model. The Compass class consists of a compass rose and a needle that indicates the current heading of RT-RCUSC. The Plot class provides a historical display of RT-RCUSC's last fifteen positions. The last position as reported by the server is at the center of the grid with numerical x and y coordinates displayed on the axis.

The Nav_Message class is responsible for building and parsing navigation messages. Data designated for transmission between the client and server are converted from integers into bytes for transmission through the Internet. Data received from the Internet are converted from bytes back into integers. The transmitted and converted data are used to update the Compass and Plot display panels.

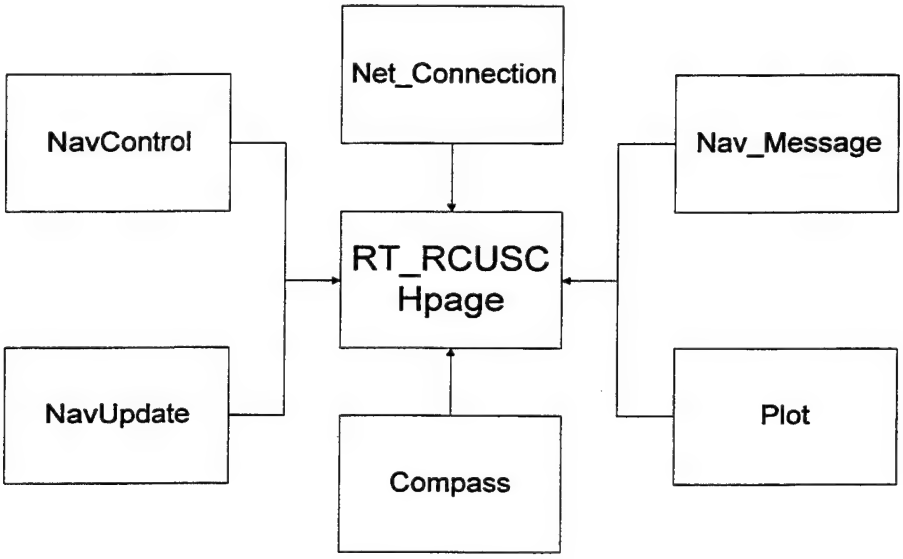


Figure 24. Class Diagram for the Client Portion of RT-RCUSC

The Net_Connection class is responsible for establishing a client socket and for communicating with the server program. The Net_Connection establishes a client socket and blocks until a socket is established. Once a socket has been established, navigation update messages are sent and received from the server. When these messages are received, the Net_Connection class sets a flag that is periodically checked by the RT_RCUSCHpage class indicating that another message is awaiting processing. Socket error conditions are captured by the Net_Connection class.

b. Java Classes for RT_Server.java

The RT_Server.java file is made up four Java classes. [Appendix J for TCP, Appendix M for UDP] The class RT_Server is the driver for RT-RCUSC's server.

This class is responsible for instantiation of the Model, Net_S_Connection and Nav_Message classes. [Fig. 25] RT_Server periodically calls the Model class requesting updates of RT-RCUSC's speed, heading and position attributes. Once RT_Server receives the updates from the Model class it then sends the data to the Nav_Message class for integer to byte conversion. Following the conversion, the data are returned to RT_Server where it controls the flow of the data to Net_S_Connection. Net_S_Connection class sends the update to the client site. RT_Server also parses messages received from the client and determines whether to update the command attributes or to stop the simulation.

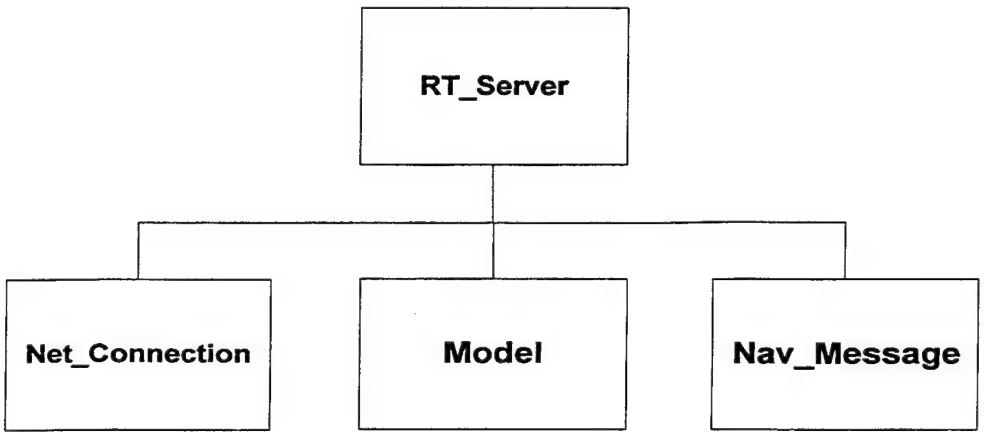


Figure 25. Class Diagram for RT-Server

The Net_S_Connection class is responsible for establishing a server socket and for communicating with the client program. The Net_S_Connection establishes a server socket and blocks until the client establishes a connection. Once a socket has been established, navigation update messages are sent and received from the client. When these messages are received, this Net_S_Connection class sets a flag that is periodically checked by the RT_Server class indicating that another message is awaiting processing. Socket error conditions are captured by the Net_S_Connection Class.

The Model class is responsible for modeling the simulated ship's attributes such as course, speed and position. This class is called periodically from RT_Server. The position and heading attributes are used to build navigation messages.

The Nav_Message class is responsible for building and parsing navigation messages. Data designated for transmission between the client and server are converted from integers into bytes for transmission through the Internet. Data received from the Internet are converted from bytes back into integers. The transmitted and converted data are used to update the Model's class command attributes.

c. Java Constants for Smart_Constants.java

The constants used by the RT_RCUSC model are contained in file Smart_Constants.java. [Appendix K for TCP, Appendix N for UDP]

B. REPORT ON THE EXPERIMENT

Initial data collection began shortly after integration testing was complete, the start date was July 6, 1997. At that time a PC which had Microsoft NT Server installed on it was unavailable to us so we executed experiments 1 and 2 from a SUN machine (white.nosc.mil) which enables us to further debug the software. This data was not included in the thesis as it was collected on equipment that did not meet the IT-21 standard. By July 12, 1997 we had secured a PC with NT Server installed in it which allowed us to execute our first two experiments.

Appendix O contains the data for experimental condition one (Server Control using TCP). Experimental condition one for TCP yielded latency mean times of 31.4 milliseconds and the greatest latency time was 60 milliseconds. [Table 1] The histogram for this data reveals a distribution composed of two modes. [Fig. 26]

Statistics for Experiment 1 (TCP)	
Mean	31.4104
Standard Error	0.558041406
Median	35
Mode	45
Standard Deviation	13.95103516
Sample Variance	194.6313821
Kurtosis	-1.207254334
Skewness	-0.21443877
Range	55
Minimum	5
Maximum	60
Sum	19631.5
Count	625
Largest(1)	60
Smallest(1)	5
Confidence Level(95.0%)	1.095866014

Table 1. Statistics for Experiment 1 (TCP)

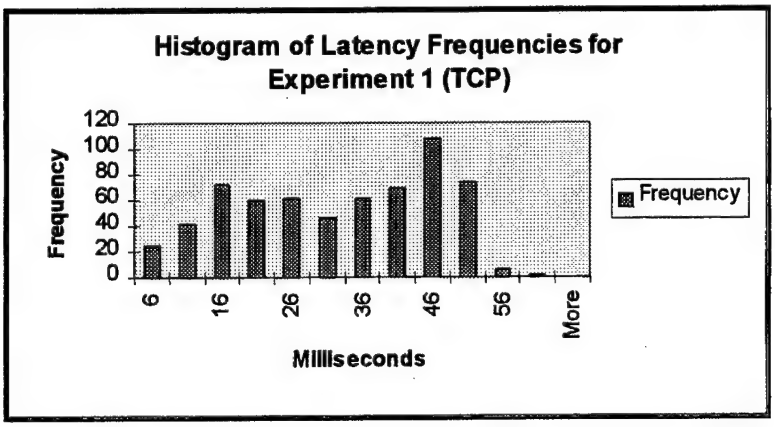


Figure 26. Histogram of Latency Frequencies for Experiment 1 (TCP)

Appendix P contains the data for experimental condition one (Server Control using UDP). Experimental condition one for UDP yielded latency mean times of 32.0 milliseconds and the greatest latency time was also 75.5 milliseconds. [Table 2] The histogram for this data reveals a distribution composed of two modes and the possibility of the formation of a third mode. [Fig. 27]

Statistics for Experiment 1 (UDP)	
Mean	32.01355932
Standard Error	0.80659356
Median	35
Mode	45
Standard Deviation	13.85369935
Sample Variance	191.9249856
Kurtosis	-0.870757961
Skewness	-0.278277581
Range	70.5
Minimum	5
Maximum	75.5
Sum	9444
Count	295
Largest(1)	75.5
Smallest(1)	5
Confidence Level(95.0%)	1.587428297

Table 2. Statistics for Experiment 1 (UDP)

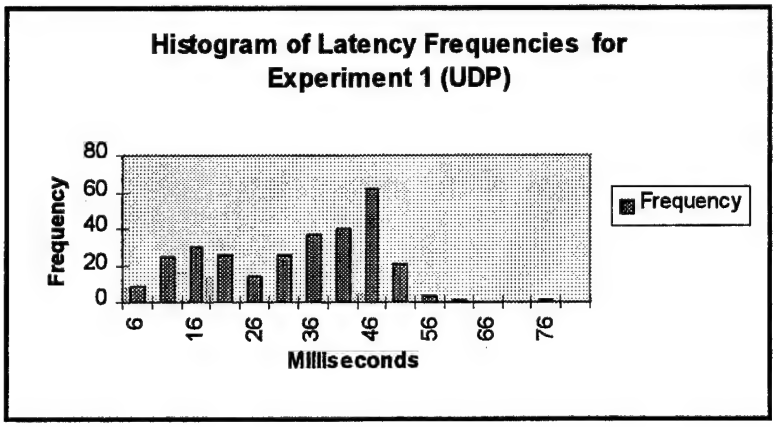


Figure 27. Histogram of Latency Frequencies for Experiment 1 (UDP)

Appendix Q contains the data for experimental condition two (Shipboard Client Control using TCP). Experimental condition two for TCP yielded latency mean times of 32.0 milliseconds and the greatest latency time was 90 milliseconds. [Table 3] The histogram for this data reveals a distribution composed of two modes and the possibility of three modes. [Fig. 28]

Statistics for Experiment 2 (TCP)	
Mean	32.0057377
Standard Error	0.659294408
Median	35
Mode	10
Standard Deviation	16.28337068
Sample Variance	265.1481608
Kurtosis	-0.814801658
Skewness	0.172826799
Range	85
Minimum	5
Maximum	90
Sum	19523.5
Count	610
Largest(1)	90
Smallest(1)	5
Confidence Level(95.0%)	1.294766769

Table 3. Statistics for Experiment 2 (TCP)

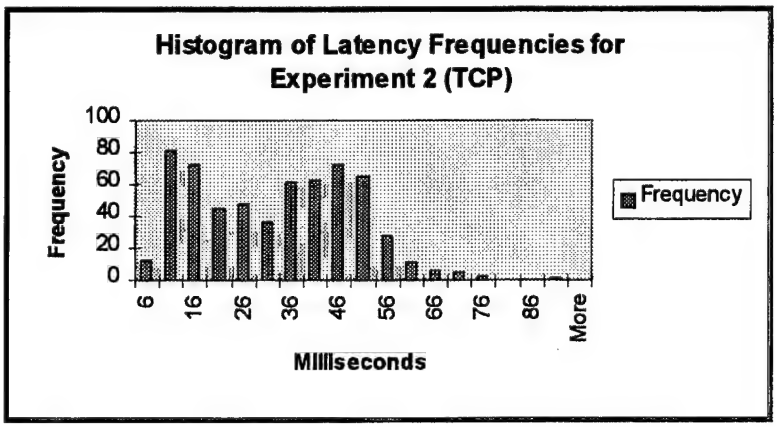


Figure 28. Histogram of Latency Frequencies for Experiment 2 (TCP)

Appendix R contains the data for experimental condition two (Server control using UDP). Experimental condition two for UDP yielded latency mean times of 31.1 milliseconds and the greatest latency time was 100.5 milliseconds. [Table 4] The histogram for this data reveals a distribution composed of two modes and the possibility of four modes. [Fig. 29]

Statistics for Experiment 2 (UDP)	
Mean	31.1271028
Standard Error	0.742804415
Median	30
Mode	10
Standard Deviation	17.1811159
Sample Variance	295.1907435
Kurtosis	-0.323314938
Skewness	0.526468684
Range	95.5
Minimum	5
Maximum	100.5
Sum	16653
Count	535
Largest(1)	100.5
Smallest(1)	5
Confidence Level(95.0%)	1.459178071

Table 4. Statistics for Experiment 2 (UDP)

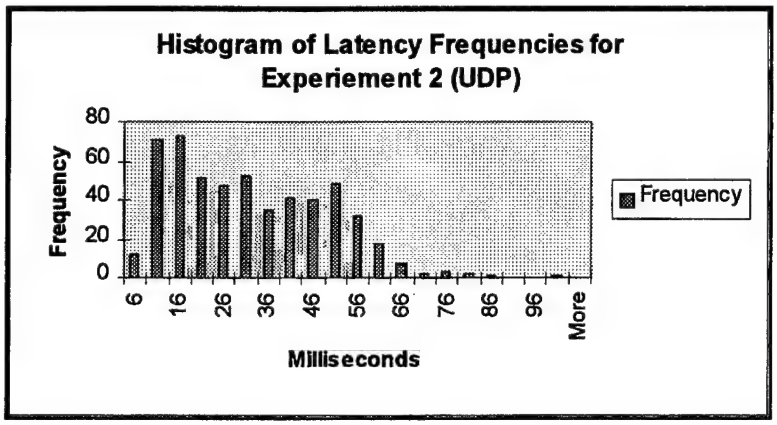


Figure 29. Histogram of Latency Frequencies for Experiment 2 (UDP)

Coincidentally the results obtained for experiments 1 and 2 using the SUN workstation as a server were nearly identical to the results obtained when using the IT-21 compliant equipment.

In all cases the histograms presented a distribution that suggested more than one mode. This indicates that collisions had occurred and retransmission was necessary.

The operational condition of the equipment and software was confirmed using a series of informal tests on July 13, 1997 in preparation for the execution of experimental conditions 3, 4 and 5 aboard the USS Coronado using a geosynchronous satellite.

On July 14 a series of tests were executed in an effort to exercise experimental condition 3. Because USS Coronado was unable to provide us access to their SIPERNET server due to security constraints and because the ship does not have a NIPERNET server we elected to run the experiment in a configuration that placed the server on land and the controlling client aboard the ship. This configuration was opposite in physical characteristics to our defined environment. Experimental condition three was intended to support the use of an Ashore Command Center that would control RT-RCUSC by attaching a client PC from the Ashore Command Center, via the Internet, to the server PC located aboard RT-RCUSC (in this case USS Coronado). Therefore without access to a shipboard server we decided to use a shipboard client and contact our land based server. It was felt that this configuration would provide identical data sets had we been able to configure the system as originally defined.

The shipboard client immediately located our server web site (http://sanchita.nosc.mil/java/tcp/nav_mod.html) and the applets were passed and began executing successfully. However after the navigation data was entered and the "START" buttonpressed the client was unsuccessful in forming a socket connection back to the server. The result was that no latency data could be obtained. This was consistent for both the TCP and UDP protocols.

An investigation into the hardware connections aboard the ship proved inconclusive as the ship's Information Services personnel felt that the problem involved either timing constraints imposed by the protocols when the protocols are used to communicate via a satellite or of because of a security check or firewall that may have been established between the client and server PCs. The IS personnel aboard Coronado indicated that if there was a firewall established aboard the ship the result would have been that we would not have reached our web site as the firewall is IP specific. Therefore the IS personnel were satisfied that no firewall was present that would have prevented us from conducting the experiment.

During the week of 14 July access to our web site was made via several other communication methods such as modems and remote Internet access through the World Wide Web and in all cases it was possible to manipulate the navigation data and collect latency data. In addition we developed another software program, "RT-Client" which was intended to execute from the client and manipulate the data on the server without the need for a web browser or HTML interface. In addition RT-Client contained logic that captured the error conditions which would allow us to determine where the fault occurred.

Upon arrival from USS Coronado data analysis was conducted on 19-20 July for the week of 14 July. In addition RT-Client was made ready for its installation aboard USS Coronado for the week of 21 July. Also on July 17 we solicited the assistance of the satellite communications group at the Naval Research and Development (NRaD) Center, San Diego, requesting access to the Navy communication satellites via an Ethernet connection. Although the Center was quite generous with their equipment, personnel and assistance bandwidth restrictions were in place by the NCTAMS eastpac because of Fleet demands which therefore eliminated this option.

On July 21 RT-Client was installed and executed aboard USS Coronado. The resultant error condition stated, "java.net.NoRouteToHostException: Host unreachable". When referring to the documentation for Java errors this condition indicated that either a firewall was present or that a router was not working. Since access to the web site was obtained both prior to and following the RT-Client run it was assumed that the router(s) were in place and functioning, therefore ruling out the possibility of a router problem. With access to the web site uninterrupted by firewall protection it was considered questionable if security constraints played a role in denying access to the server data.

Subsequent discussions with Dr. Luqi on July 26 resulted in our generation of an E-Mail to Professor Volpano requesting assistance. Indications are that the programs have been developed properly and an explanation for the problem remains unknown.

In an attempt to perform the experiments contacts were made to colleagues in Australia, Guam and Puerto Rico. Latency times obtained from the remote execution of the program from Australia indicated a mean time of 192 milliseconds. This value is

significantly lower than that expected if communications were conducted via satellite as the propagation delay for end-to-end transit time for a geosynchronous satellite transmission is 270 milliseconds. [Ref. 22] As a result of this test we have learned that distance alone is not the proper criteria in assuring that geosynchronous satellite communications are invoked when using the Internet.

In addition to Australia, Guam and Puerto Rico we attempted to contact the USS Constellation (CV-64) through their Public Affairs Office in an effort to request their participation in our experiment, unfortunately we were unable to make contact with personnel able to assist.

On August 8, 1997 we conducted an experiment to reinforce our findings that the USS Coronado is not configured to accept Java socket connections. Indications were that when USS Coronado was in port that its connection to the Internet was via an Ethernet connection provided by a pier side cable connections. When we discovered that we could access the Internet from the ship but were prevented from making a socket connection we felt confident that it was a security setting disallowing a socket connection. Returning to NRaD we tested the pathway to USS Coronado by "pinging" a computer aboard the USS Coronado. We discovered that the latency time was 641 milliseconds and the trace indicated that the NCTAMS was included, possibly indicating that a satellite path was used. Certainly the latency time suggests a satellite path was used. Therefore we could not prove that the problem was caused by security restrictions aboard the ship or timing problems within the communications path.

To date we have been unsuccessful in gathering latency data for experiment 3 and in executing experimental conditions 4 and 5 using geosynchronous satellites.

The testing of experimental conditions 3-5 using LEO satellites became a possibility when on June 3, 1997 Mr. Jerry Neuner, of AIRINC, offered LEO satellite transmission/reception equipment for our thesis work. AIRINC is a license for the LEO satellite transmission/reception equipment built by ORBCOMM which is the company that owns and operates its own LEO satellite constellation. Mr. Neuner contacted ORBCOMM on my behalf to make the arrangements for the equipment delivery. Mr. Stan Young of ORBCOMM contacted me directly concerning the use and application of

the equipment and upon his approval the equipment was to be delivered. On July 10, I contacted Mr. Neuner to determine the status of the equipment. Mr. Neuner indicated that ORBCOMM had approved my request for the equipment but had not delivered it to AIRINC who would immediately forward it on to me. In addition Mr. Neuner, who had offered technical assistance in modifying our software in order for it to execute using the ORBCOMM equipment, requested a copy of the software. I immediately provided him copies of our program.

Since July 10 I have contacted Mr. Neuner twice by phone and have been told that the equipment is, "on its way".

Without the ORBCOMM equipment in our possession we are unable to perform our LEO satellite data collection and analysis.

Based on the logistical issues outlined above we feel that we have exhausted our options in obtaining the support necessary in order to complete all aspect of the experimental data collection necessary to bring this topic to closure.

V. SUMMARY AND CONCLUSION

A. EVALUATION OF THE MODEL

The model was found to be useful in the collection of latency timing data. The data was used to perform a variety of statistical tests and to develop histograms which were used to evaluate the timing constraints of a real-time command and control system that communicates from client to host using the Internet. The model's effectiveness was never fully realized because of restricted availability of both geosynchronous and LEO satellites.

B. UNRESOLVED ISSUES

Due to the restricted availability of both geosynchronous and LEO satellites it was not possible for us to complete experimental conditions 3, 4 and 5. At such time when those restrictions are lifted the experiments should be conducted and the data analyzed. The results of the analysis should be presented and used a guideline for the latency considerations when trying to built a C4ISR system that requires remote control through the Internet.

C. SUMMARY

Real-time, remote control of an unmanned surface combatant using the Internet is a capability that is within our present day technological grasp. It was our hope that our experimental conditions would have allowed for the collection of latency data that would have provided the critical timing constraints associated with the use of the Internet for such control. The timing constraints could then be incorporated into the architectural considerations for systems such as RT-RCUSC. Unfortunately our research fell short of it goal.

In reviewing the latency data that was collected for both experimental conditions one and two, using either TCP or UDP protocols, our results reinforce the premise that real-time control is not only possible but practical when controlling a system directly from the server or via a client connected through a Local Area Network (LAN). With latency values substantially below the 100 millisecond level constructing a real-time system is certainly achievable.

It was very disappointing for us when we came to the realization that we would be unable to collect the latency data for experimental conditions three, four and five.

Unidentified problem located within the Navy's geosynchronous satellite communications network as well as limited access to the Navy's satellite communications equipment restricted our ability to complete our geosynchronous data collection.

Logistical problems involving the release of commercial LEO satellite communications equipment eliminated our ability to gather our LEO satellite data.

The final value of this effort may offer a limited contribution to our colleagues in the discipline of Software Engineering. Yet if such an assignment's true value is judged by the enlightenment and knowledge gained by the students who have pursued this topic then it has achieved the loftiest of goals. The level of integration required in order to conduct these experiments has been a substantial challenge which has opened many new doorways both intellectually as well as interpersonally.

Certainly RT-RCUSC is possible. Social and political influences may restrict the use of a RT-RCUSC in the immediate future however based on the technological trend it is obvious that this technology will be refined and implemented with greater intensity in the near future.

Of constant concern is the question of security. Work in the areas of network security is addressing these concerns and therefore it seems reasonable to assume that the technology will develop rapidly.

Vulnerability of such systems to electronic jamming is certainly possible however communication schemes that provide encryption and frequency shifting seem likely to solve those issues.

LIST OF REFERENCES

1. SMART SHIP *Home Page*, http://www.dt.navy.mil/smartship/smartship.html

2. Luqi, Valdis Berzins, *Rapidly Prototyping Real-Time Systems*, IEEE Software, September 1988, pp. 25-36.

3. Luqi, *Software Evolution Through Rapid Prototyping*, IEEE Software, May, 1989, p. 18.

4. Boehm, Barry W., Terrance E. Grey, Thomas Seewaldt, *Prototyping Verses Specifying: A Multiproject Experiment, IEEE Transactions on Software Engineering*, Vol. SE-10, No. 3, May 1984.

5. *Pentagon Spy Plane Crashes*, Military News Service, 2/26/97.

6. *Teledyne Ryan Unveils USAF's Global Hawk UAV*, Military News Service, 2/21/97.

7. *Kamnski Assured Outrider Development to be Completed in May*, Military News Service, 2/19/97.

8. *Miniature Planes Set for Battlefield Role*, Military News Service, 2/13/97.

9. *DARPA looks at role of UAVs in Cruise Missile Defense*, Military News Service, 3/6/97.

10. SELF DEFENSE TEST SHIP (SDTS) *Home Page*, http://www.nswses.navy.mil/~lev/sdts.html

11. *Arsenal Ship, Jane's Picture of the Week*, http://www.arpa/asjpo

12. Blazar, Ernest. *Future Shock / Arsenal Ship Will Have Small Crew & Big Punch.* Navy Times 07-29-96 Issue. http://www.arpa.mil/asjpo/ntimes4.html

13. Jane's Picture of the Week, http://www.thomson.com/janes/jfs9602.html

14. Walman, Lt. Jon P. Walman, *The 'New Generation' of Combatants SC21, Arsenal Ship Reflect 21stCentury Vision.* http://kraken.nwscc.sea06.navy.mil/sc21/html/surfwar.htm12

15. Mission Need Statement for a 21st Centurary Tactical Aviation Sea-Based Platform(U), http://www.navsea.navy.mil/cvx/cvxmns.html

16. *Stealth Warship Plans Launched,* Military News Service 10/26/96.

17. *Boeing Picked for 'Internet in the Sky'*, Military News Service, 5/01/97.

18. *The Acceleration of World Wide Wireless Communications.* The Seventh Annual Symposium Sponsored by the Center for Telecommunications at Stanford and Accel Partners, May 21, 1997.

19. Grant, Conrad, J. *Real Time Definition.* The Navy's Program Executive Office, Theater Air Defense' Combat System Functional Allocation Board, 7 May, 1997.

20. Selic, Bran, Garth Gullekson, and Paul T. Ward. *Real-Time Object-Oriented Modeling.* New York, New York: John Wiley & Sons, Inc., 1994.

21. Tanenbaum, Andrew, S. *Computer Networks.* Upper Saddle River, New Jersey: Prentice Hall, 1996.

APPENDIX A. IT-21 DEFINITION

This Naval message, distributed by the Commander's in Charge of the Pacific and Atlantic Fleets (CINCPACFLT, CINCLANTFLT), provides the Information Technology for the 21st (IT-21) hardware/software implementation standards for programs installing information systems on Fleet units/bases and provides the Fleet with guidance on maintaining existing information systems until the installation of IT-21 products.

RATUZYUW RUEOMCB9916 0891106-UUUU--RUWFOAA.
ZNR UUUUU ZUI RUHPSGG9842 0890945 ZUI RUEOMCB6600 0891100
ZUI RUEOMCB6613 0891103
RHHJJAA T JICPAC HONOLULU HI
RHNVDDD T USS SUPPLY
RHOOMIU T MIUWU ONE ONE THREE
RHRMCAA T NAVCOMTELSTA BAHRAIN
RUCTFOA T USCGC KEY LARGO
RUHBABA T CG THIRD MARDIV
RUHBANA T MTCC OKINAWA JA
RUHDYOK T NAVSECGRUACT YOKOSUKA JA
RUWNAVL T USS JOHN S MCCAIN
RUWNAWQ T USS MCCLUSKY
R 300944Z MAR 97 ZYB PSN 077925Q30
FM CINCPACFLT PEARL HARBOR HI//N00//
TO ALPACFLT
ALLANTFLT
INFO RUENAAA/ASSTSECNAV RDA WASHINGTON DC//C4I//
RUENAAA/CNO WASHINGTON DC//N00/N09/N095/N2/N4/N41/N43/N46/N6/N6B/
N8/N80/N85/N86/N87/N88//
RUCBCLF/CINCLANTFLT NORFOLK VA//N00/N6//
RUCBACM/CINCUSACOM NORFOLK VA//J00/J6//
RHHMUNA/USCINCPAC HONOLULU HI//J00/J6//
RHDLCNE/CINCUSNAVEUR LONDON UK//00/N6//
RULSSEA/COMNAVSEASYSCOM WASHINGTON DC//N00/N08/PMS335/PMS3
RUENMED/BUMED WASHINGTON DC//N00//
RHRMDAB/RUCJNAV/COMUSNAVCENT//N00/N6//
RUCTPOA/CNET PENSACOLA FL//N00//
RUEACNP/BUPERS WASHINGTON DC//N00//
RUHEHMS/COMMARFORPAC//CG/G6//
RUCKMAA/COMMARFORLANT//CG/G6//
RULSSPA/COMSPAWARSYSCOM WASHINGTON DC//N00/N05/PMW171/PMW176//
RUWFADO/NAVSTKAIRWARCEN FALLON NV//N00//
RULKSDF/COMNAVSECGRU FT GEORGE G MEADE MD//N00//
RULSAMX/COMNAVSUPSYSCOM MECHANICSBURG PA//N00//
RUWFAFK/COMNAVSPECWARCOM CORONADO CA//N00//
RULSDG/NRL WASHINGTON DC//N00//
RULSWCB/COMNAVCOMTELCOM WASHINGTON DC//N00/N3//
RUCOSAO/NAVMASSO CHESAPEAKE VA//N00/N6//
RUWFOAA/NCCOSC RDTE DIV SAN DIEGO CA/N433//

RHHMHAH/CINCPACFLT PEARL HARBOR HI//N00//
BT
UNCLAS //N05230//
ALPACFLT 008/97
MSGID/GENADMIN/CINCPACFLT/008//
SUBJ/INFORMATION TECHNOLOGY FOR THE 21ST CENTURY//
POC/M.R. SCOTT/CDR N6/CINCPACFLT/-/TEL: 808 471-8637//
POC/D.A. STRAUB/CDR N6/CINCLANTFLT/-/TEL: 757 322-5863//
RMKS/1. THIS IS THE FIRST IN A SERIES OF JOINT CINCPACFLT AND CINCLANTFLT MESSAGES CONCERNING THE DEVELOPMENT AND IMPLEMENTATION OFIT-21. THIS MESSAGE PROVIDES IT-21 HARDWARE/SOFTWARE IMPLEMENTATION STANDARDS FOR PROGRAMS INSTALLING INFORMATION SYSTEMS ON FLEET UNITS/BASES AND PROVIDES THE FLEET WITH GUIDANCE ON MAINTAINING EXISTING INFORMATION SYSTEMS UNTIL INSTALLATION OF IT-21 PRODUCTS. THE IT-21 IMPLEMENTATION STANDARDS OUTLINED BELOW ARE PROMULGATED IN ADVANCE OF DON-WIDE GUIDANCE FROTHE DON CHIEF INFORMATION OFFICER (CIO). THE DON CIO WILL PROMULGATE DON-WIDE STANDARDS FOLLOWING NEGOTIATION OF ENTERPRISE-WIDE NETWORK OPERATING SYSTMS AND APPLICATIONS.
2. BACKGROUND: INFORMATION SUPERIORITY IS THE FOUNDATION OF JOINT VISION 2010 BATTLEFIELD DOMINANCE, AS WELL AS THE WARFIGHTING VISION FOR EACH SERVICE. NETWORK WARFARE, ROBUST INFRASTRUCTURE AND INFORMATION DISSEMINATION TO DISPEED FORCES ARE KEY ELEMENTS IN ACHIEVING INFORMATION SUPERIORITY. IT-21 IS A FLEET DRIVEN REPRIORITIZATION OF C4I PROGRAMS F RECORD TO ACCELERATE THE TRANSITION TO A PC BASED TACTICAL/TACTICAL SUPPORT WARFIGHTING NETWORK. THE INACTIVATION OF THE CURRENT DOD MESSAGING SYSTEM (AUTODIN) BY DEC 99, WITH NO PLANNED NAVY INFRASTRUCTURE REPLACEMENT, MANDAT THE RAPID IMPLEMENTATION OF THIS WARFIGHTING NETWORK.
3. COMMERCIAL NETWORK OPERATING SYSTEMS (NOS) AND E-MAIL PRODUCTS HAVE ACHIEVED FUNCTIONAL PARIT. THE FLEETS CANNOT CONTINUE TO SUPPORT A MULTITUDE OF DIVERSE OPERATING SYSTEMS AND E-MAIL PRODUCTS WITH THEIR OWN TRAINING, OPERATIONAL PROCEDURES AND TROUBLESHOOTING REQUIREMENTS. THE DOD JOINT TECHNICAL ARCHITECTURE (JTA) AND DEFENSE INFORMATION INFRTRUCTURE COMMON OPERATING ENVIRONMENT (DII COE) PROVIDE DOD WITH THE AIS SYSTEM GUIDANCE REQUIRED TO TAKE THE NAVY INTO THE 21ST CENTURY. THIS CONVERGENCE OF SOLUTIONS, PROBEMS AND GUIDANCE PROVIDES THE IMPETUS TO ESTABLISH MINIMUM NAVY AIS STANDARDS AT THIS TIME. IMPLEMENTATION OF THIS POLICY REQUIRES ALL NON-STANDARD NOS AND E-MAIL PRODUCTS BE REPLACED NLT DEC 99.
A. WINDOWS NT SERVER 4.0 IS THE STANDARD FLEET NOS. IT WILL SOON BE FOLLOWED BY WINDOWS NT 5.0. WINDOWS NT SERVER 4.0 IS DII COE COMPLIANT.
B. MS EXCHANGE IS DESIGNATED AS THE STANDARD E-MAIL SOLUTION FOR BOTH FLEETS TO ENSURE AN INTEROPERABLE SECURE MESSAGING SYSTEM IS OPERATIONAL PRIOR TO AUTODIN INACTIVATION NLT DEC 99.
C. MS OFFICE 97 IS DESIGNATED AS THE STANDARD FLEET OFFICE SOFTWARE.
D. EXPENDITURE OF OPERATING FUNDS TO MAINTAIN EXISTING IT-21 NONCOMPLIANT NOS AND APPLICATIONS SHALL BE THE ABSOLUTE MINIMUM NECESSARY TO MEET OPERATING REQUIREMENTS UNTIL IT-21 NOS/SOFTWARE IS INSTALLED EVEN IF TEMPORARY LAN DEGRADATION OCCURS. SOFTWARE

REQUIREMENTS DRIVE HARDWARE STANDARDS. HARDWARE AND SOFTWARE PURCHASED TODAY MUST BE CAPABLE OF MEETING MISSION REQUIREMENTS THROUGH THE YEAR 2000.

4. CINCPACFLT AND CINCLANTFLT ARE ACTIVELY WORKING WITH OPNAV ON IT-21 FUNDING AND IMPLEMENTATION PLANS. IN GENERAL, AFLOAT IT-21 IMPLEMENTATION WILL BE LINKED TO DEPLOYING BATTLEGROUPS AND ASHORE IT-21 WILL BE IMPLEMENTED IN A PHASED APPROACH. SPECIFIC IMPLEMENTATION SCHEDULES WILL BE PROMULGATED AT A LATER DATE. CINCPACFLT AND CINCLANTFLT ARE TRANITIONING TO WNDOWS NT 4.0, MS EXCHANGE AND MICROSOFT OFFICE 97. THIS ENVIRONMENT CANNOT BE OPTIMIZED WITHOUT 32 BIT OPERATING SYSTEMS, HIGH RESOLUTION DIPLAYS AND MASS STORAGE. ATM BACKBONE LANS WITH AT LEAST 100 MBS (TCP/IP)TO THE DESKTOP PC WILL BE INSTALLED ON ALL SHIPBOARD LANS, FLEET HEADQUARTERS (CPF, CLF, TYCOMS, GROUPAND SQUADRON MMANDS) AND SHOULD BE INSTALLED IN THOSE SHORE ACTIVITIES THAT SUPPORT TACTICAL OPERATIONS. THIS WILL THEN ALLOW TRANSITION TOATM-TO-THE-DESKTOP PC WHN THE ATM TECHNOLOGY MATURES.

5. SYSTEM COMMANDS AND PROGRAM MANAGERS:

A. NTCSS WILL BECOME THE IT-21 PROGRAM O RECORD FOR INSTALLATION OF BOTH SECRET AND UNCLASSIFIED LANS ONBOARD COMMISSIONED SHIPS. NTCSS (ATIS/SNAP III) LANS INSTALLED FROM THIS POINT ON WLL HAVE AN ATM BACKBONE, 100 MBS (FAST ETHERNET) TO THE DESKTOP PC AND THE HARDWARE/SOFTWARE OUTLINED AT THE END OF THIS MESSAGE. THE MIGRATION OF NTCSS LANS TO HIGHER CAPACITY LANS WILL REDUCE THE NUMBER OF PC'S DELIVERED DURING INITIAL INSTALLATION. THE TRADE-OFF OF QUANTITY FOR FRONT END PC'S IS REQUIRED TO SUPPORT JV-2010 AND AUTODIN INACTIVATION.

B. SPAWAR IS WORKING WITH NAVSEA TO ENSURE THAT LANS INSTALLED DURING NEW CONSTRUCTION MEET THE IT-21 REQUIREMENTS.

C. APPLICATION PROGRAM MANAGERS SUCH AS JMCIS, NSIPS, TAMPS, AND GCSS SHOULD MIGRATE CURRENT APPLICATIONS TO THE DII COE WITH AN IMMEDIATE OBJECTIVE OF OBTAINING PC WORKSTATION ACCESS TO ALL APPLICATION DATA ON AN ENTERPRISE LAN.

D. PROGRAMS INSTALLING INFORMATION SYSTEMS (NEWNET, SMARTLINK, SMARTBASE, TELEMEDICINE, ETC.) MUST INSTALL COMPONENTS IN FLEET ACTIVITIES THAT MEET IT-21 STANDARDS AND PROVIDE INTEROPERABILITY THROUGHOUT THE WARFIGHTING NETWORK.

6. TYCOMS AND THIRD ECHELON COMMANDS SHALL ENSURE THAT:

A. SHIPS AND ACTIVITIES INSTALLING NEW LANS, UNDERGOING SIGNIFICANT LAN UPGRADES OR THOSE ACTIVITIES WITH STAND ALONE PC'S SHALL INSTALL IT-21 HARDWARE AND SOFTWARE. NEW OR REPLACEMENT SHIPBOARD AND SHORE BASED TACTICAL LANS SHOULD HAVE AN ATM BACKBNE WITH AT LEAST 100 MBS (FAST ETHERNET) TO THE PC.

B. SHIPS AND ACTIVITIES WITH EXISTING LANS, WHICH REQUIRE REPLACEMENT OF UNSERVICEABLE HARDWARE, SORT OF A FULL NETWORK UPGRADE, SHALL INSTALL HARDWARE WHICH MEETS IT-21 STANDARDS. THE NEW EQUIPMENT MAY NOT BE COMPATIBLE WITH THE EXISTING LAN HARDWARE. CINCPACFLT AND CINCLANTFLT BELIEVE THAT ALL AUTOMATED INFORMATION SYSTEMS (AIS) PROCURED MUST BE COMPATIBLE WITH THE IT-21 LAN STANDARDS EVEN IF TEMPORARY LAN DEGRADATION OCCURS. THERE IS ONLY SUFFICIENT FUNDING TO DO IT RIGHT THE FIRST TIME.

7. THE IT-21 STANDDS BELOW REPRESENT FRONT END MARKET TECHNOLOGY, ARE DYNAMIC IN NATURE, AND WILL CONTINUE TO BE CLOSELY LINKED TO COMMERCIAL TENDS. THE STANDARDS LISTED BELOW ARE INTENDED TO BE

MINIMUM STANDARDS AND WILL BE UPDATED PERIODICALLY.
A. IT-21 LAN:
(1) AFLOAT LAN STANDARDS - ATM FIBER BACKBONE, 100 MBPS
//
R 300944Z MAR 97 ZYB PSN 077825Q29
UNCLAS //N05230//
FINAL SECTION OF 02
ALPACFLT 008/97
MSGID/GENADMIN/CINCPACFLT/008//
SUBJ/INFORMATION TECHNOLOGY FOR THE 21ST CENTURY//
RMKS/
(FAST ETHERNET) TO THE PC.
(2) ASHORE TACTICAL AND HEADQUARTERS COMMAND CENTER STANDARD (CPF, CLF, TYCOMS, GROUP AND SQUADRON COMMANDS) - ATM BACKBONE, 100 MBPS (FAST ETHERNET) TO THE PC.
(3) ASHORE TACTICAL SUPPORT COMMAND STANDARDS (BASES) - ATM
=20
=20
=20
PAGE 03 RUHPSGG9916 UNCLAS
BACKBONE, 100 MBPS (FAST ETHERNET) TO THE PC.
(4) METROPOLITAN AREA NETWORKS (MAN) SHOULD BE CAPABLE OF SUPPORTING AT LEAST OC-3 (155MBS).
B. IT-21 SOFTWARE:
- WINDOWS NT 4.0/5.0 WORKSTATION
- MS OFFICE 97 PROFESSIONAL (WORD 97, POWERPOINT 97, EXCEL 97, S ACCESS 97)
- IBM ANTI VIRUS (NAVY LICENSE, AVAIL FROM NAVCIRT)
- MS BACK OFFICE CLIENT
- MS OUTLOOK 97
- MS EXCHANGE 5.0
- MS IMAGE COMPOSER
C. IT-21 DATABASES. RELATIONAL DATABASES THAT CAN SUPPORT WEB TECHNOLOGY IAW THE COE (ORACLE, SYBASE, SQL SERVER, ACCESS, ETC.) WILL BE USED TO SUPPORT DATA REQUIREMENTS AND APPLICATION DEVELOPMENT. ALL PROCESS ENGINEERING INITIATIVES THAT RESULT IN DESIGN/REDESIGN OF A DATA COLLECTION/CAPTURE SYSTEM MUST USE COE COMPLIANT RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS) SOFTWARE. THIS REQUIREMENT IS PROVIDED TO ENSURE RDBMS INITIATIVES USE COTS APPLICATION SOFTWARE. FOR ADDITIONAL INFORMATION ON RELATIONAL
=20
=20
=20
PAGE 04 RUHPSGG9916 UNCLAS
DATABASES CONTACT CDR SANDY BUCKLES, CPF N67, COMM/DSN (808) 474-6384, NIPRNET U67@CPF-EMH.CPF.NAVY.MIL.
D. MINIMUM IT-21 PC CAPABILITIES: CPF CAN CURRENTLY PURCHASE THE IT-21 STANDARD PC WITH SOFTWARE FOR $3250.00 - $3579.00 - SEE PARA 7(H) AND 7(I).
- 200 MHZ PENTIUM PRO CPU
- 64 MB EDO RAM
- 3.0 GB HARD DRIVE
- 3.5 INCH FLOPPY DISK DRIVE
- 8X IDE CD-ROM

- DUAL PCMCIA/PC CARD READER
- PCI VIDEO W/2MB RAM
- 17 INCH MONITOR (1280 X 1024)
- POINTING DEVICE (TRACKBALL OR MOUSE)
- SOUNDBLASTER (COMPATIBLE) AUDIO CARD WITH SPEAKERS KEYBOARD
- CPU COMPATIBLE 100 MBPS FAST ETHERNET NIC

E. STANDARD IT-21 LAPTOP WORKSTATION: APPROXIMATELY $5300 - SEE PARA 7(H).

- 150 MHZ PENTIUM
- 32 MB EDO RAM

=20
=20
=20

PAGE 05 RUHPSGG9916 UNCLAS

- 12.1 IN SVGA ACTIVE MATRIX COLOR DISPLAY
- 2.1 GB EIDE HDD
- 6X INTERNAL CD-ROM
- MODEM, PCMCIA SLOTS, NIC CARD
- SMART LITHIUM BATTERY

F. IT-21 NT FILE SERVER FOR DIRECTORY NETWORK SERVICE: APPROXIMATELY $26K - SEE PARA 7(H). THESE ARE MINIMUM SPECIFICATIONS. NEEDS OF THE SPECIFIC NETWORK WILL DICTATE REQUIREMENTS.

- DUAL 166 MHZ PENTIUM CPU
- 512K SECONDARY CACHE MEMORY- 256 MB RAM
- TWO 4 GB SCSI HDD
- ONE 6 GB DAT DRIVE
- ONE 3.5 INCH FLOPPY DISK DRIVE
- 6X SCSI CD-ROM
- DUAL PCMCIA/PC CARD READER
- 2 DPT SCSI III CACHING CONTROLLERS (SMARTCACHE 4)
- PCI VIDEO W/2MB RAM
- 17 INCH MONITOR (1280 X 1024)
- POINTING DEVICE (TRACKBALL OR MOUSE)

=20
=20
=20

PAGE 06 RUHPSGG9916 UNCLAS

- KEYBOARD
- TWO CABLETRON CPU COMPATIBLE ATM NIC CARDS
- ANTEC DUAL POWER SUPPLY CASE (HOT SWAPPABLE)

G. IT-21 FILE SERVER/APPLICATION SERVER: APPROXIMATELY $26K - SEE PARA 7(H). SAME AS IT-21 NT FILE SERVER FOR DIRECTORY NETWORK SERVICE WITH THE FOLLOWING CHANGES:

- CHANGE HDD RQRMT TO FIVE 4 GB DRIVES
- CHANGE DAT TO 18 GB.

H. PRICES FOR PC TECHNOLOGY ARE CONSTANTLY CHANGING AND CAN VARY GREATLY DEPENDING ON METHOD OF PROCUREMENT. FOR EXAMPLE, ON 28 MAR 97 AN IT-21 PC PURCHASED DIRECTLY FROM A VENDOR COSTS $3643. GOVERNMENT RATE FOR SMALL PURCHASES (LESS THAN TEN) IS $3579. A BULK PROCUREMENT (MORE THAN SEVENTY-FIVE) COSTS $3250. THE ABOVE PRICES INCLUDE SHIPPING. BULK PROCUREMENTS SHOULD BE MADE THROUGH THE TYPE COMMANDERS WHEN APPROPRIATE. MR. RICK KOOKER, CPF N65, COMM/DSN:(808) 474-5882, NIPRNET: U65@CPF-EMH.CPF.NAVY.MIL IS

AVAILABLE TO ASSIST TYCOMS WITH AIS PROCUREMENT ISSUES.
I. AS NETWORK COMPUTER TECHNOLOGY EVOLVES SOME COMMANDS MAY BE ABLE TO TRANSITION TO NETWORK COMPUTERS. WHEN CONSIDERING INSTALLATION OF NETWORK COMPUTERS, TOTAL NETWORK COST MUST BE
=20
=20
=20
PAGE 07 RUHPSGG9916 UNCLAS
EVALUATED. NETWORK COMPUTERS HAVE NOT MATURED SUFFICIENTLY TO IMPLEMENT THEM IN FLEET PLATFORMS AT THIS TIME.
8. WAIVER REQUESTS FROM THE ABOVE STANDARDS SHOULD BE SUBMITTED DIRECTLY TO THE RESPECTIVE CPF/CLF N6. POINTS OF CONTACT ARE AS FOLLOWS:
A. CINCLANTFLT: CDR DEBRA STRAUB AT COMM (757) 322-5863, NIPRNET: U6@CLF.NAVY.MIL
B. CINCPACFLT: CDR MIKE SCOTT AT COMM (808) 474-7860, NIPRNET:U6@CPF-EMH.CPF.NAVY.MIL.//

BT#9916

APPENDIX B. MAIN.HTML

```
<HTML>
<HEAD>
<TITLE>RT-RCUSC Home Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+3>
RT-RCUSC (TCP)
</font></b></p>
    The Real-Time, Remotely Controlled, Unmanned, Surface Combatant (RT-RCUSC)
is the exploration of using Internet connections and protocols for control
and communication between an unmanned ship and various controlling sites.  See
<a href="THS_SUM.HTML">thesis description</a> for more information.
<a href="sm4.html">Thesis demonstration</a> brings up a WEB site of options
that are prototypes developed in this thesis used to collect timing information.
</BODY>
</HTML>
```

APPENDIX C. THS_SUM.HTML

```
<HTML>
<HEAD>
<TITLE>Thesis Abstract Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+2>
RT-RCUSC Thesis Abstract (TCP)
</font></b></p>
<APPLET CODE="Thesis_Sum.class" WIDTH=980 HEIGHT=1000><APPLET>
</BODY>
</HTML>
```

APPENDIX D. SM4.HTML

```
<HTML>
<HEAD>
<TITLE> Option Selection Page (TCP) </TITLE>
</HEAD><BODY>
<P align-center><b>
<font size=+2>
RT-RCUSC Model Options (TCP)
</font></b></p>
 <ul>
  <li><a href="nav_mod.html">Navigation Model</a>
  <li><a href="rad_mod.html">Radar and IFF Model</a>
  <li><a href="wea_mod.html">Weapons Model</a>
 </ul>
</BODY>
</HTML>
```

APPENDIX E. NAV_MOD.HTML

```
<HTML>
<HEAD>
<TITLE>Navigation Module Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+2>
RT-RCUSC SCRIPTED NAVIGATION CONTROL MODEL (TCP)
</font></b></p>
<APPLET CODE="RT_RCUSCHpage.class" WIDTH=980 HEIGHT=500><APPLET>
</BODY>
</HTML>
```

APPENDIX F. RAD_MOD.HTML

```
<HTML>
<HEAD>
<TITLE>RADAR & IFF Module Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+2>
RT-RCUSC RADAR & IFF CONTROL MODEL  (TCP)
</font></b></p>
        UNDER CONSTRUCTION
</BODY>
</HTML>
```

APPENDIX G. WEA_MOD.HTML

```
<HTML>
<HEAD>
<TITLE>Weapons Module Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+2>
RT-RCUSC WEAPONS CONTROL MODEL (TCP)
</font></b></p>
        UNDER CONSTRUCTION
</BODY>
</HTML>
```

APPENDIX H. THESIS_SUM.JAVA

```
import java.awt.*;
// Class Thesis_Sum contains a summary of the thesis.
// This class is executed when thesis description is
// selected from the root web page.
public class Thesis_Sum extends java.applet.Applet
 {
  public void init()
   {
    setLayout(new GridLayout(2,1,10,10));
    Panel panel1 = new Panel();
    add(panel1);
    final String summary =
" This thesis was developed in response to the Navy's  goal to reduce staffing levels \n" +
" aboard surface combatants. The thesis describes the computers, peripherals, and  \n" +
" communication networks that make a Real-Time, Remotely Controlled,  \n" +
" Unmanned, Surface Combatant, (RT-RCUSC) possible using wire and wireless  \n" +
" Internet connections and protocols.  \n" +
"  A Command and Control (C2) model was developed using the rapid prototype\n" +
" methodology. The C2 model collected latency data which was analyzed to  \n" +
" determine the feasibility of a RT-RCUSC.  \n" +
"  Sixteen experiments using latency times were designed to determine the  \n" +
" viability of communication paths that progressively increased in distance and  \n" +
" complexity. Variables included the use of two protocols, TCP and UDP, the use  \n" +
" of two satellite types, geosynchronous and Low Earth Orbiting (LEO),as well as  \n" +
" employing up to two satellites per end-to-end transmission path.  \n" +
"  The results demonstrated that real-time control of a ship's navigation system  \n" +
" can be performed when entries are made directly on the server PC or when using \n" +
" a client PC that is connected to the server PC via an Ethernet LAN. When  \n" +
" controlling RT-RCUSC directly from the server using TCP and UDP the mean  \n" +
" latency time was approximately 31.4 and 32.0 milliseconds respectively with  \n" +
" the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly\n" +
" when controlling RT-RCUSC from a client, connected to the server via a LAN,  \n" +
" using TCP and UDP, the mean latency time was approximately 32.0 and 31.1  \n" +
" milliseconds respectively with the greatest latency time equal to 90 and 100.5  \n" +
" milliseconds respectively. Security restrictions prevented Java socket formation  \n" +
" between the client/server interface when testing wireless paths aboard the USS  \n" +
" Coronado. The restrictions prevented us from gathering latency data for our  \n" +
" geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite  \n" +
" communications transmission/reception equipment failed to be provided for our  \n" +
" evaluation. Therefore we were unable to gather latency data for our LEO  \n" +
" wireless connection experiments.  \n" +
"  Future research needs to focus on gathering latency data from both  \n" +
```

```
" geosynchronous and LEO satellites for the purposes of determining the viability  \n" +
" of a RT-RCUSC in order to determine the effectiveness of wireless communications \n"
" with respect to Naval C2 system. \n";
panel1.add(new TextArea(summary,20,60));
  } // End of init

                                                    }                                                    //End of Thesis_Sum
```

APPENDIX I. RT_RCUSCHPAGE.JAVA FOR TCP

```
// RT_RCUSCHpage.java is the client portion of the RT_RCUSC software.  It is made
// up of seven classes.  RT_RCUSCHpage is the driver of the client program and is
// responsible for instantiation of the Navcontrol, Net_Connection. Nav_message
// NavUpdate, Compass and Plot classes.

import java.awt.*;
import java.net.*;
import java.io.*;


// Class RT_RCUSCHpage - Main class of this applet.
//  RT_RCUSCHpage is the driver of the client portion
//  of RT_RCUSC program.
//  It's responsible for instansiation of the four
//  supporting objects - NavControl, NavUpdate, Compass,
//  and Plot.
/** RT_RCUSCHpage.java
* @version 1.1
* @author Floyd Bailey
* Implementation for RT_RCUSCHpage main control
*/

/**
 * Constructor for class RT_RCUSCHpage.
 */

public class RT_RCUSCHpage extends java.applet.Applet
 implements Runnable, Smart_Constants
{
        // VARIABLES for RT_RCUSCHpage
        // init_nav             - NavControl object declaration
        // update_nav           - NavUpdate object declaration
        // compass_update       - Compass object declaration
        // plot_update          - Plot object declaration
        // ship_model           - Model object declaration
        // first_time           - Boolean variable indicating whether
        //                      -   initial course and speed was entered
        // i                    - Integer variable used for exception
        //                      -   handler
        // InitThread           - Thread object declaration
        // MAX_ARRAY            - Integer constant used to determine the
        //                      -   size of array
        // point_history        - Array of Point used to be last MAX_ARRAY
        //                      - x,y position of RT_RCUSC
        // temp_point           - Temporary Point variable
        // array_index          - Integer variable indicating the number of
        //                      -   elements in point_history
        // ship_info            - Integer array used to update RT_RCUSC
        //                      -   position parameters
        // cmd_info             - Integer array used to update RT_RCUSC
```

```
//                      -   command parameters
// ret_value           -  Boolean variable assign returning condition
//                      -   of methods returning boolean type
// panel1              -  Instansiation of Java Panel class

NavControl init_nav;
NavUpdate update_nav;
Compass compass_update;
Plot plot_update;
Nav_Message nav_msg;          // Class to handle building and parsing nav
Net_Connection net_con;       // Class to handle network connection
static boolean first_time = true;
int i = 0;
Thread InitThread;
static Point[] point_history = new Point[MAX_ARRAY];
Point temp_point,temp_point2;
static int array_index = 0;
static int[] ship_info = new int[4];
static int[] cmd_info = new int[3];
static int update_trigger = 0;      //Used to trigger update every 2 sec
boolean ret_value;
String host;
static byte[] msg_buffer = new byte[MAX_MESSAGE_LENGTH];
byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH];
Panel panel1 = new Panel();
byte record_type;.
```

```
/** Method init() of class SmartShip - This method instansiates objects
 *  of classes NavUpdate, Compass, Plot and NavUpdate.
*/
 public void init()
  {
   host = this.getCodeBase().getHost();
   init_nav = new NavControl(this, "LAT", "LON", "X_POS", "Y_POS", "COURSE", "SPEED",
                             0, 0, 0, 0, 0, 0);
   update_nav = new NavUpdate(this, "COURSE", "SPEED", host, 0, 0, 0);
   compass_update = new Compass(this);
   plot_update = new Plot(this);
   setBackground(Color.white);
   setLayout(new GridLayout(2,2,10,10));
   add(init_nav);
   add(update_nav);
   add(compass_update);
   add(plot_update);
                //Get address of applet's home
  }             //end of init

/** Method insets(int,int,int,int) of class SmartShip - This method
 *  determines the spaces between the four panels - init_nav,
 *  update_nav, compass_update and plot_update.
```

```
*/
 public Insets insets()
  {
   return new Insets(10,10,10,10);
  }       // end of insets

/** Method start() of class SmartShip - This method instansiates thread
 * InitThread and starts it.  This thread updates the panel and takes
 * inputs from the user.
*/
 public void start()
  {
   if (InitThread == null)
    {
     InitThread = new Thread(this);
     InitThread.start();
   }                  //End of == null
  }                   //End of start

/** Method stop() of class SmartShip - this method stops thread
 *  InitThread and deinstansiates it.
*/
 public void stop()
  {
   if (InitThread != null)
    {
     InitThread.stop();
     InitThread = null;
     net_con.stop();          //Disconnect connection
   }                          // End of != null
  }                           //End of stop




/** Method run() of class SmartShip - This method is the main loop of the
 *   class.  This is where data and control is exchanged between the five
 *   supporting objects - init_nav, update_nav, compass_update, plot_update
 *   and ship_model.
*/
 public void run()
  {
   while (true)          // Loop forever
    {
     if (init_nav.running())
      {
       if (first_time)  // Get SmartShip initial speed and course
        {
         try
          {
           ret_value = init_nav.init_info(ship_info);
            nav_msg = new Nav_Message(ship_info[0],ship_info[1],ship_info[2],ship_info[3]);
```

```
    nav_msg.get_nav_msg(msg_buffer);   //Get msg to send
    net_con = new Net_Connection(InetAddress.getByName(host));
    net_con.send_msg(msg_buffer);      //Send out init position to server
        first_time = false;
   }
  catch (Exception e)
   {
   System.out.println(e);
   }          // End of catch
 }            // End of first_time
else          // SmartShip initial data already entered
 {
 if (update_trigger >= 23)
  {                              // Get SmartShip command speed, course and acceleration update
   update_trigger = 0;
   cmd_info[0] = 25;              // Speed
   cmd_info[1] = cmd_info[1] + 3;        // Course
   if (cmd_info[1] > 359)
    cmd_info[1] = 0;
   nav_msg.update_msg(cmd_info);      // Nav_msg class builds update msg
   nav_msg.get_nav_msg(msg_buffer); // Get msg to send
   net_con.send_msg(msg_buffer);       // Send msg to server
  }                                    // End of if update_command
 if (net_con.data_receive())          // Has server sent info
  {
   net_con.clear_data_receive();
   net_con.get_message(rec_buffer);
   record_type = rec_buffer[0];
   switch (record_type)                // Parse server info
    {
     case NAV_TYPE :
      nav_msg.get_ship_info(rec_buffer,ship_info);
      rec_buffer[REC_SUB_TYPE_POS] = NAV_STAT;
      net_con.send_msg(rec_buffer);  //Send msg back to server for timing
      break;
     default:// Shouldn't be getting here
    }                  // End of switch logic
   ret_value = compass_update.set_compass(ship_info[2]);
  }                   // End of if data_receive
 if (array_index < MAX_ARRAY)
  {
   temp_point = new Point(ship_info[0],ship_info[1]);
   point_history[array_index] = temp_point;
   array_index++;
  }          // End of if array not full
 else
  {
   temp_point2 = point_history[MAX_ARRAY - 1];
   for (int for_index = (MAX_ARRAY - 1); for_index > 0; for_index--)
    {
     temp_point = point_history[for_index - 1];
      point_history[for_index - 1] = temp_point2;
      temp_point2 = temp_point;
    }                  // End of for loop
```

```
          temp_point = new Point(ship_info[0],ship_info[1]);
          point_history[MAX_ARRAY - 1] = temp_point;
         }                    // End of else array not full yet
        ret_value = plot_update.plot_array(point_history,array_index);
        repaint();
        compass_update.repaint();
        plot_update.repaint();
        }                     // End of else Smart ship data already entered
       try
        {
         Thread.sleep(100); // One tenth of second delay between positon updates
         update_trigger++;
        }                     // End of  try
       catch (InterruptedException e) { } // Sleep exception
      }                       // End of if running
     repaint();
    }                         // End of while (true) loop
  }                           // End of run

/** Method update(NavControl) of class RT_RCUSC */
void update(NavControl Nav_in)
 {
    //Future code can go here
 }                            // End of update NavControl .

/** Method update(NavUpdate) of class RT_RCUSC - TBD */
void update(NavUpdate Nav_in)
 {
                              // Future code can go here
 }                            // End of update NavUpdate


/** Method paint(Graphics)   */
 public void paint(Graphics g)
  {
   g.setColor(Color.red);
   g.drawString("        RUCUS ",0, 60);
  }
 }

/** Class NavControl  - Supporting class of RT_RCUSCHpage.
 *  It's responsible for initial SmartShip data
*/

class NavControl extends Panel
 {
        // VARIABLES for class NavControl
        // running          - Boolean variable which indicates if object is
        //                    running
        // lat              - TextField variable to output latitude data to screen
        // lon              - TextField variable to output longitude to screen
        // x_pos            - TextField variable to output x position to screen
        // y_pos            - TextField variable to output y position to screen
        // course           - TextField variable to output course to screen
```

```
        // speed          - TextField variable to output speed to screen
        // r_lat          - Integer variable for current latitude of SmartShip
        // r_lon          - Integer variable for current longitude of SmartShip
        // r_x_pos        - Integer variable for current x position of SmartShip
        // r_y_pos        - Integer variable for current y position of SmartShip
        // r_icourse      - Integer variable for current SmartShip heading
        // r_ispeed       - Integer variable for current SmartShip speed
        // MAX_SPEED      - Constant maximum speed of SmartShip
        // MIN_SPEED      - Constant minimum speed of SmartShip
        // MAX_CRS        - Constant maximum allowable compass input
        // MIN_CRS        - Constant minimum allowable compass input
        // MIN_X          - Constant minimum allowable x axis value
        // MAX_X          - Constant maximum allowable x axis value
        // MIN_Y          - Constant minimum allowable y axis value
        // MAX_Y          - Constant maximum allowable y axis value
        // MAX_LAT        - Constant maximum allowable latitude value
        // MIN_LAT        - Constant minimum allowable latitude value
        // MAX_LON        - Constant maximum allowable longitude value
        // MIN_LON        - Constant minimum allowable longitude value
        // start_button   - Constant String value for push button label
        // stop_button    - Constant String value for pust button label
        // outerparent    - Variable used to declare RT_RCUSCHpage as parent
        //                  to NavControl

static boolean running = false;
static TextField lat,lon,x_pos,y_pos,course,speed;
static int r_lat = 0;
static int r_lon = 0;
static int r_x_pos = 0;
static int r_y_pos = 0;
static int r_icourse = 0;
static int r_ispeed = 0;
final int MAX_SPD = 45;
final int MIN_SPD = -20;
final int MAX_CRS = 360;
final int MIN_CRS = 0;
final int MIN_X = -1024;
final int MAX_X = 1024;
final int MAX_Y = 1024;
final int MIN_Y = -1024;
final int MAX_LAT = 90;
final int MIN_LAT = -90;
final int MIN_LON = -180;
final int MAX_LON = 180;
final String start_button = "START";
final String stop_button = "STOP";
RT_RCUSCHpage outerparent;

 /** NavControl  is the Constructor for class NavControl. It takes 12
   * parameters 6 of type String and 6 parameters of double. NavControl
   * is where initial ship's attributes are entered. The user also enters the start
  *and stop commands here.
  * @param parm1 string label for latitude of RT_RCUSC.
  * @param parm2 string label for longitude of RT_RCUSC.
```

```
   * @param parm3 string label for x position of RT_RCUSC.
   * @param parm4 string label for y position of RT_RCUSC.
   * @param parm5 string label for course of RT_RCUSC.
   * @param parm6 string label for speed of RT_RCUSC.
   * @param flt1 initial Latitude of RT_RCUSC.
   * @param flt2 initial Longitude of RT_RCUSC.
   *@param flt3 initial x position of RT_RCUSC 0.
   * @param flt4 initial y position of RT_RCUSC 0.
   * @param flt5 initial course of RT_RCUSC.
   * @param flt6 initial speed of RT_RCUSC.
   */
NavControl(RT_RCUSCHpage target,String parm1, String parm2, String parm3,
        String parm4, String parm5, String parm6, double flt1, double flt2,
        double flt3, double flt4, double flt5, double flt6)
{
 this.outerparent = target;
 setLayout(new GridLayout(7,2,10,10));  //Sets up a grid of 7 rows and 2 columns

 // Convert float parameters to string
 lat = new TextField(String.valueOf(flt1),6);
 lon = new TextField(String.valueOf(flt2),6);
 x_pos = new TextField(String.valueOf(flt3),6);
 y_pos = new TextField(String.valueOf(flt4),6);
 course = new TextField(String.valueOf(flt5),6);
 speed = new TextField(String.valueOf(flt6),6);



 //Set edit boxes to a white background
 lat.setBackground(Color.white);
 lon.setBackground(Color.white);
 x_pos.setBackground(Color.white);
 y_pos.setBackground(Color.white);
 course.setBackground(Color.white);
 speed.setBackground(Color.white);

 //Create label and text fields
 add(new Label(parm1, Label.RIGHT));
 add(lat);
 add(new Label(parm2, Label.RIGHT));
 add(lon);
 add(new Label(parm3, Label.RIGHT));
 add(x_pos);
 add(new Label(parm4, Label.RIGHT));
 add(y_pos);
 add(new Label(parm5, Label.RIGHT));
 add(course);
 add(new Label(parm6, Label.RIGHT));
 add(speed);

 // Create and label two buttons
 add(new Button("START"));
 add(new Button("STOP"));
```

```
 setBackground(Color.green);
 }      //End of constructor NavControl

 /** Method insets sets up the spacing between items in this panel - labels, text boxes
  *  and buttons.
 */
 public Insets insets()
  {
  return new Insets(10,10,10,150);
  }      //End of insets

 /** Method init_info loads initial latitude, longitude, course and
  *   speed into an array.
 */
 public boolean init_info(int info[])
  {
  info[0] = r_lat;
  info[1] = r_lon;
  info[2] = r_icourse;
  info[3] = r_ispeed;
  return true;
  }      //End of init_info
```

```
/** Method action(Event,Object) handles selection of buttons
 *   and TextFields on this panel.
 * @param evt type of event to trigger this action
 * @param arg object type causing this action
*/
 public boolean action(Event evt, Object arg)
  {
  if (evt.target instanceof TextField)
   {
    outerparent.update(this);
    return true;
   }              //End of if true
  else if (evt.target instanceof Button)
   {
    String label = (String)arg;
    if (label.equals(start_button))
     {
      read_values();
      return true;
     }             // End of if START button was selected
    else
     {
```

```
        running = false;
        return true;
      }             // STOP button was selected
    }
   else
    return false;
  }             //End of action

/** Method running returns a boolean type indicating what
 *    state this panel is in - running or not running.
*/
public boolean running()
 {
  return running;
 }      //End of running

/** Method read_values reads and checks for values entered by the user.
*/
public void read_values()
 {
        // VARIABLES for method read_values
        // crs_string       -  String variable used to store course data
        // spd_string       -  String variable used to store speed data
        // xpos_string      -  String variable used to store x position data
        // ypos_string      -  String variable used to store y position data
        // lat_string       -  String variable used to store latitude data
        // lon_string       -  String variable used to store longitude data
        // crs_good         -  Boolean variable indicating proper course entry
        // spd_good         -  Boolean variable indicating proper speed entry
        // xpos_good        -  Boolean variable indicating proper x position entry
        // ypos_good        -  Boolean variable indicating proper y position entry
        // lat_good         -  Boolean variable indicating proper latitude entry
        // lon_good         -  Boolean variable indicating proper longitude entry
        // tmp_spd          -  Integer variable used for computations
        // tmp_crs          -  Integer variable used for computations
        // tmp_xpos         -  Integer variable used for computations
        // tmp_ypos         -  Integer variable used for computations
        // tmp_lat          -  Integer variable used for computations
        // tmp_lon          -  Integer variable used for computations
 String crs_string;
 String spd_string;
 String xpos_string;
 String ypos_string;
 String lat_string;
 String lon_string;
 boolean crs_good;
 boolean spd_good;
 boolean xpos_good;
 boolean ypos_good;
 boolean lat_good;
 boolean lon_good;
 int tmp_spd = r_ispeed;
 int tmp_crs = r_icourse;
 int tmp_xpos = r_x_pos;
```

```
int tmp_ypos = r_y_pos;
int tmp_lat = r_lat;
int tmp_lon = r_lon;

// Next 7 instructions assigns values enter by user into String variables
crs_string = course.getText();
spd_string = speed.getText();
xpos_string = x_pos.getText();
ypos_string = y_pos.getText();
lat_string = lat.getText();
lon_string = lon.getText();

// Initialize data as valid.
crs_good = spd_good = xpos_good = ypos_good = lat_good = lon_good = true;

// Following try block parses user data and checks for correct range and type.
try
 {
  tmp_crs = Integer.parseInt(crs_string);
  if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
   {
    crs_good = false;
    course.setBackground(Color.red);
        course.setText(String.valueOf(r_icourse));
  }             // End of failed course entry
 tmp_spd = Integer.parseInt(spd_string);
 if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
  {
   spd_good = false;
   speed.setBackground(Color.red);
   speed.setText(String.valueOf(r_ispeed));
  }   // End of failed speed entry
 tmp_xpos = Integer.parseInt(xpos_string);
 if ((tmp_xpos > MAX_X) || (tmp_xpos < MIN_X))
  {
   xpos_good = false;
   x_pos.setBackground(Color.red);
   x_pos.setText(String.valueOf(r_x_pos));
  }             // End of failed x pos entry
 tmp_ypos = Integer.parseInt(ypos_string);
 if ((tmp_ypos > MAX_Y) || (tmp_ypos < MIN_Y))
  {
   xpos_good = false;
   y_pos.setBackground(Color.red);
   y_pos.setText(String.valueOf(r_y_pos));
  }             // End of failed y pos entry
 tmp_lat = Integer.parseInt(lat_string);
 if ((tmp_lat > MAX_LAT) || (tmp_lat < MIN_LAT))
  {
   lat_good = false;
   lat.setBackground(Color.red);
   lat.setText(String.valueOf(r_lat));
  }             // End of failed lat entry
 tmp_lon = Integer.parseInt(lon_string);
```

```
    if ((tmp_lon > MAX_LON) || (tmp_lon < MIN_LON))
     {
      lon_good = false;
      lon.setBackground(Color.red);
      lon.setText(String.valueOf(r_lon));
     }   // End of failed lon entry
    }              // End of try
   catch (Exception e)
    {

    }              // End of catch
   if ((crs_good == true) && (spd_good == true) && (xpos_good == true) &&
         (ypos_good == true) && (lat_good == true) && (lon_good == true))
    {
    course.setBackground(Color.white);
    speed.setBackground(Color.white);
    x_pos.setBackground(Color.white);
    y_pos.setBackground(Color.white);
    lat.setBackground(Color.white);
    lon.setBackground(Color.white);
    course.setText(String.valueOf(tmp_crs));
    r_icourse = tmp_crs;
    speed.setText(String.valueOf(tmp_spd));
    r_ispeed = tmp_spd;
    x_pos.setText(String.valueOf(tmp_xpos));
    r_x_pos = tmp_xpos;
    y_pos.setText(String.valueOf(tmp_ypos));
    r_y_pos = tmp_ypos;
    lat.setText(String.valueOf(tmp_lat));
    r_lat = tmp_lat;
    lon.setText(String.valueOf(tmp_lon));
    r_lon = tmp_lon;
    running = true;
   }              // End of good values submitted
  }               // End of read_values
 }                // End of class NavControl



/** Class NavUpdate is where command data is enter by the user
 *  after the start of a run.
*/
 class NavUpdate extends Panel
  {
        // VARIABLES for class NavUpdate
        // course        - String variable used to store course data
        // speed         - String variable used to store speed data
        // acceleration  - String variable used to store acceleration data
        // update_cmd    - Boolean variable indicating if command data was
        //               -  updated
        // outerparent   - Object variable indicating that RT_RCUSCHpage is
         //              -  parent to this class
```

```
        // r_spd          - Integer variable indicating current command speed
        // r_crs          - Integer variable indicating current command course
        // r_acl          - Integer variable indicating current command acceleration
        // MAX_SPD        - Constant integer, maximum speed for SmartShip
        // MIN_SPD        - Constant integer, minimum speed for SmartShip
        // MAX_CRS        - Constant integer, maximum course (degrees ) for course
        // MIN_CRS        - Constant integer, minimum course (degrees ) for course
        // MAX_ACL        - Constant integer, maximum acceleration for SmartShip
        // MIN_ACL        - Constant integer, minimum acceleration for SmartShip
static TextField course,speed,acceleration;
static boolean update_cmd = false;
RT_RCUSCHpage outerparent;
static int r_spd = 0;
static int r_crs = 0;
static int r_acl = 0;
final static int MAX_SPD = 45;
final static int MIN_SPD = -20;
final static int MAX_CRS = 360;
final static int MIN_CRS = 0;
final static int MAX_ACL = 6;
final static int MIN_ACL = -6;

/** Constructor for class NavUpdate - takes 6 parameters.  3 of type String and
 *     3 of type Integer.
 * @param target
 * @param parm1 string type used as a label for Course text field.
 * @param parm2 string type used as a label for Speed text field.
 * @param parm3 string type used as a label for Acceleration text field.
 * @param flt1 integer type used for course enter by user.
 * @param flt2 integer type used for speed enter by user.
 * @param flt3 integer type used for acceleration enter by user.
*/
NavUpdate(RT_RCUSCHpage target,String parm1, String parm2, String parm3,
          int flt1, int flt2, int flt3)
{
 this.outerparent = target;
 setBackground(Color.blue);
 // Next instruction sets up a grid for this panel - 4 rows by 2 colums.
 setLayout(new GridLayout(4,2,10,40));
 String blank = "      "; //Used to align submit button

 // Convert float parameters to string
 course = new TextField(String.valueOf(flt1),5);
 speed = new TextField(String.valueOf(flt2),5);
 acceleration = new TextField(String.valueOf(flt3),5);

 // Set TextField background to white.
 course.setBackground(Color.white);
 speed.setBackground(Color.white);
 acceleration.setBackground(Color.white);

 // Create label and text fields
 add(new Label(parm1, Label.RIGHT));
 add(course);
```

```
 add(new Label(parm2, Label.RIGHT));
 add(speed);
 add(new Label(parm3, Label.RIGHT));
 add(acceleration);
 add(new Label(blank, Label.RIGHT));
 add(new Button("SUBMIT"));
}                //End of constructor NavUpdate

/** Method insets determines spaces between item in this panel - TextFields, labels
*    and button.
*/
public Insets insets()
 {
 return new Insets(10,10,10,70);
 }      //End of insets

/** This method get_cmd_data(int cmd_info[]) the current command data.
 * @param cmd_info[] is an integer array containing command data
 *    The values are read into an array which the
 *    caller can use to update SmartShip position.
*/
public boolean get_cmd_data(int cmd_info[])
 {
 cmd_info[0] = r_spd;
 cmd_info[1] = r_crs;
 cmd_info[2] = r_acl;
 return true;
 }                // end of get_cmd_data

/** Method action handles TextField and button action of this panel.
 * @param evt is of type Event, it's the event of this action.
 * @param arg is of type Object, it's the object of this action.
*/
public boolean action(Event evt, Object arg)
 {
 if (evt.target instanceof TextField)
  {
   outerparent.update(this);
   return true;
  }               // End of if true
 else if (evt.target instanceof Button)
  {
  read_values();
  update_cmd = true;
  return true;
  }
 else
  return false;
 }                // End of action

/** Method update_command() returns a boolean indicating if command
 *    data has been updated.
*/
public boolean update_command()
```

```
{
 return update_cmd;
}                          // End of update_command


/** Method clear_update_cmd() set update_cmd to false.
*/
public void clear_update_cmd()
 {
  update_cmd = false;
 }                         // End of clear_update

/** Method read_values() validates and assigns command data entered by user.
*/
public void read_values()
 {
        // VARIABLES for method read_values()
        // crs_string      -   String variable used to store course data
        // spd_string      -   String variable used to store speed data
         // acl_string     -   String variable used to store acceleration data
        // crs_good        -    Boolean variable indicating valid course entry
        // spd_good        -   Boolean variable indicating valid speed entry
        // acl_good        -   Boolean variable indicating valid acceleration
        // tmp_spd         -   Integer used for computation
        // tmp_crs          -   Integer used for computation
        // tmp_acl .        -   Integer used for computation
 String crs_string;
 String spd_string;
 String acl_string;
 boolean crs_good;
 boolean spd_good;
 boolean acl_good;
 int tmp_spd;
 int tmp_crs;
 int tmp_acl;

 // Next 3 instruction assigns command data to String variables.
 crs_string = course.getText();              // Get text of course text box
 spd_string = speed.getText();               // Get text of speed text box
 acl_string = acceleration.getText();        // Get text of acceleration text box

 tmp_spd = r_spd;
 tmp_crs = r_crs;
 tmp_acl = r_acl;

 // Initialize valid data to true.
 crs_good = spd_good = acl_good = true;
 try
  {
  tmp_crs = Integer.parseInt(crs_string);
  if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
   {
    crs_good = false;
    course.setBackground(Color.red);
```

```
        course.setText(String.valueOf(r_crs));
      }             // End of failed course entry
     tmp_spd = Integer.parseInt(spd_string);
     if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
      {
       spd_good = false;
       speed.setBackground(Color.red);
       speed.setText(String.valueOf(r_spd));
      }             // End of failed speed entry
     tmp_acl = Integer.parseInt(acl_string);
     if ((tmp_acl > MAX_ACL) || (tmp_acl < MIN_ACL))
      {
       acl_good = false;
       acceleration.setBackground(Color.red);
       acceleration.setText(String.valueOf(r_acl));
      }             // End of failed acceration entry
    }               // End of try
   catch (Exception e)
    {

    }               // End of catch
   if ((crs_good == true) && (spd_good == true) && (acl_good == true))
    {
     course.setBackground(Color.white);
     speed.setBackground(Color.white);
     acceleration.setBackground(Color.white);
     course.setText(String.valueOf(tmp_crs));
     r_crs = tmp_crs;
     speed.setText(String.valueOf(tmp_spd));
     r_spd = tmp_spd;
     acceleration.setText(String.valueOf(tmp_acl));
     r_acl = tmp_acl;
    }               // End of good values submitted
  }                 // End of read_values

}                   // End of class NavUpdate

/** Class compass displays the current SmartShip heading.
*/
class Compass extends Canvas
{
        // VARIABLES for class Compass
        // COMPASS_X_CEN      - Constant integer compass x center
        // COMPASS_Y_CEN      - Constant integer compass y center
        // MAX_Y              - Constant integer used for drawing needle
        // MAX_X              - Constant integer used for drawing needle
        // MAX_LIT_Y          - Constant integer used for drawing needle
        // degree             - Double variable, current compass value
        // degs_rads          - Double variable, radian value of the heading
        // x_values           - Integer array used to draw needle
        // y_values           - Integer array used to draw needle
        // pts                - Integer variable, number of points to draw needle
        // rad_to_deg         - Double variable, conversion factor of degrees to rads
        // outerparent        - Variable indicating RT_RCUSCHpage as parent
```

```
        // cos_ret                  - Double variable, cosine of heading
        // sin_ret          - Double variable, sine of heading

 final int COMPASS_X_CEN = 130;
 final int COMPASS_Y_CEN = 110;
 final int MAX_Y = 50;
 final int MAX_LIT_Y = 5;
 final int MAX_X = 10;
 static double degree = 30.0;
 double degs_rads;
 int x_values[] = {140,130,120,130,140};
 int y_values[] = {110,60,110,113,110};
 int pts = x_values.length;
 double rad_to_deg = Math.PI / 180.0;
 RT_RCUSCHpage outerparent;
 double cos_ret,sin_ret;

 /** Constructor Compass(SmartShipHpage) creates an object of class Compass
  * @param target is of type RT_RCUSCHpage
 */
 Compass(RT_RCUSCHpage target)
  {
   this.outerparent = target;
   setBackground(Color.blue);
  }               // End of Compass

/** Method set_compass(int) sets new heading for compass.
 * @param compass_value is of type int, sets compass to new value.
*/
public boolean set_compass(int compass_value)
  {
   degree = (double) compass_value;
   return true;
  }               // Set_compass

 /** Method paint(Graphics) displays current heading.
  * @param g is of type Graphics, use to paint to screen.
 */
 public void paint(Graphics g)
  {
   degs_rads = degree * rad_to_deg;
   cos_ret = Math.cos(degs_rads);
   sin_ret = Math.sin(degs_rads);
   x_values[0] = (int) (Math.round(cos_ret * MAX_X) + COMPASS_X_CEN);
   y_values[0] = (int) (Math.round(sin_ret * MAX_X) + COMPASS_Y_CEN);
   x_values[1] = (int) (Math.round(sin_ret * MAX_Y) + COMPASS_X_CEN);
   y_values[1] = (int) (COMPASS_Y_CEN - Math.round(cos_ret * MAX_Y));
   x_values[2] = (int) (COMPASS_X_CEN - Math.round(cos_ret * MAX_X));
   y_values[2] = (int) (COMPASS_Y_CEN - Math.round(sin_ret * MAX_X));
   x_values[3] = (int) (COMPASS_X_CEN - Math.round(sin_ret * MAX_LIT_Y));
   y_values[3] = (int) (COMPASS_Y_CEN + Math.round(cos_ret * MAX_LIT_Y));
   x_values[4] = x_values[0];
   y_values[4] = y_values[0];
   // Next instruction instansiates a new polygon object - the needle
```

```
   Polygon comp_needle = new Polygon(x_values,y_values,pts);
   g.setColor(Color.white);
   g.fillOval(50,30,160,160);
   g.drawString("360",120,25);
   g.drawString("180",120,205);
   g.drawString("270",31,116);
   g.drawString("90",212,116);
   g.setColor(Color.black);
   g.drawLine(130,30,130,190);
   g.drawLine(50,110,210,110);
   g.fillPolygon(comp_needle);
  }                // End of paint
}                  // End of class Compass

/** Class Plot plots current (at grid center) and up to the last 15
 *   SmartShip positions.
*/
class Plot extends Canvas
{
        // VARIABLES for class Plot
        // outerparent        -  Variable that indicates that RT_RCUSCHpage is parent
        // MAX_ARRAY     -  Constant int - maximum number of points that can be plotted
        // ship_history       -  Array of type points - positions of SmartShip
        // last_point         -  Number of positions in ship_history
        // x_center           -  String variable used to display x value of grid center
        // y_center           -  String variable used to display y value of grid center

 RT_RCUSCHpage outerparent;
 static final int MAX_ARRAY = 25;
 static Point[] ship_history = new Point[MAX_ARRAY];
 static int last_point = 0;
 String x_center,y_center;

 /** Constructor Plot(RT_RCUSCHpage) used to create an instance of class Plot.
  * @param target is of type RT_RCUSCHpage
 */
 Plot(RT_RCUSCHpage target)
 {
  this.outerparent = target;
  setBackground(Color.green);
 }                 // End of Compass

 /** Method plot_array(Point[], int) updates SmartShip positions array.
  * @param ship_hist[] of type Point array - last positions of RT_RCUSC.
  * @param last_entry of type int - is the number of postions in array.
 */
 public boolean plot_array(Point ship_hist[], int last_entry)
 {
        // VARIABLES for method plot_array
        // index          - Integer variable used as array index
        // index2         - Integer  variable used as array index
        // keep_looping   - Boolean variable used for loop control

  int index;
```

```
 int index2;
 boolean keep_looping = true;
 index2 = (last_entry - 1);
 index = 0;
 while (keep_looping)
  {
   ship_history[index] = ship_hist[index2];
   if ((index2 <= 0) || (index >= (MAX_ARRAY - 1)))
    keep_looping = false;
   else
    {
     index++;
     index2--;
    }              // End of else
  }                // End of while keep_looping
 last_point = index;
 return true;
}                  // End of plot_array

/** Method paint(Graphics) plots SmartShip positions to panel.
*/
public void paint(Graphics g)
 {
        // VARIABLES for method paint(Graphics)
        // REC_WIDTH          - Constant integer, grid width
        // REC_LENGTH         - Constant integer, grid length
        // index              - Integer variable used for array index
        // prev_x             - Integer variable used for line computations
        // prev_y             - Integer variable used for line computations
        // cur_x              - Integer variable used for line computations
        // cur_y              - integer variable used for line computations

 final int REC_WIDTH = 180;
 final int REC_LENGTH = 180;
 int index,prev_x,prev_y,cur_x,cur_y;
 prev_x = 0;
 prev_y = 0;
 g.setColor(Color.white);
 g.fillRect(50,15,REC_WIDTH,REC_LENGTH);
 g.setColor(Color.black);
 g.drawLine(140,15,140,195);
 g.drawLine(50,105,230,105);
 x_center = Integer.toString(ship_history[0].x);
 y_center = Integer.toString(ship_history[0].y);
 g.drawString(x_center,130,12);
 g.drawString(y_center,20,105);
 for (index = 0; index <= last_point; index++)
  {
   if (index > 0)
    {
     if ((Math.abs(ship_history[index].x) < REC_WIDTH) &&
       (Math.abs(ship_history[index].y) < REC_LENGTH))
     {
      cur_x = ship_history[index].x - ship_history[0].x;
```

```
        cur_x = -1 * cur_x;
        cur_y = ship_history[index].y - ship_history[0].y;
        g.setColor(Color.green);
        g.drawLine((140 + prev_y),(105 + prev_x),(140 + cur_y),(105 + cur_x));
        g.setColor(Color.black);
        prev_x = cur_x;
        prev_y = cur_y;
      }          // End of boundary check
    }            // End of if > 0
  }              // End of for loop
 }               // End of paint
}                // End of class Plot
// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants
{
// VARIABLES for class Nav_Message
// MSG_LENGTH              - Max length for nav message is 36 bytes
// NUM_OF_SHIP_PARMS       - Nav message contains 4 parameters for ship
// LAT_PARM_POS            - Latitude parameter position in byte buffer
// LON_PARM_POS            - Longitude parameter position in byte buffer
// CRS_PARM_POS            - Course parameter position in byte buffer
// SPD_PARM_POS            - Speed parameter position in byte buffer
// Y_POS                   - Ship's y position in byte buffer
// X_POS                   - Ship's x position in byte buffer
// CRS_POS                 - Ship's course position in byte buffer
// SPD_POS                 - Ship's speed position in byte buffer
// C_CRS_POS               - Ship's command course position in byte buffer
// C_SPD_POS               - Ship's command speed position in byte buffer
// C_ACL_POS               - Ship's command acceleration in byte buffer
// PARM_LENGTH             - Parameter's length - 4 bytes
// ZERO_BYTE               - Byte of 0's
// index2                  - Used for loop variable
// ser_number              - Serial number of message
// MAX_SERIAL              - Largest serial number after which value is set to 0
// nav_data                - Byte buffer used for storing messages
// char_digit              - Character used to store a digit as a character
 final static int MSG_LENGTH = 36;
 final static int NUM_OF_SHIP_PARMS = 4;   //x pos,y pos,course and speed
 final static int LAT_PARM_POS = 4;
 final static int LON_PARM_POS = 8;
 final static int CRS_PARM_POS = 12;
 final static int SPD_PARM_POS = 16;
 final static int X_POS = 4;
 final static int Y_POS = 8;
 final static int CRS_POS = 12;
 final static int SPD_POS = 16;
 final static int C_CRS_POS = 4;
 final static int C_SPD_POS = 8;
 final static int C_ACL_POS = 12;
 final static int PARM_LENGTH = 4;
 final static byte ZERO_BYTE = 0;
 static byte[] nav_data = new byte[MSG_LENGTH];
 char char_digit;
```

```
/** Nav_Message Constructor for class
*/
Nav_Message()
 {

 }                  // Constructor with no arguments

/** Nav_Message(int,int,int,int) Constructor for class - 4 parameters
 * @param srt_lat is starting latitude of RT_RCUSC.
 * @param srt_lon is starting longitude of RT_RCUSC.
 * @param srt_crs is starting course of RT_RCUSC.
 * @param srt_spd is starting speed of RT_RCUSC.
*/
Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
 {
  int index,index2;
  int num_length,num_128,rem_128;
  for (index = 0; index < MSG_LENGTH; index++)
    nav_data[index] = ZERO_BYTE;          // Zero out message array

  // Code to fill in speed field of nav message
  nav_data[SPD_PARM_POS + 3] = (byte)(0x0000007f & srt_spd);

  // Code to fill in course field of nav message
  num_128 = srt_crs / 128;
  nav_data[CRS_PARM_POS + 2] = (byte)num_128;
  rem_128 = srt_crs % 128;
  nav_data[CRS_PARM_POS + 3] = (byte)rem_128;
  nav_data[MSG_TYPE_POS] = NAV_TYPE;
  nav_data[REC_TYPE_POS] = NAV_INIT;
  nav_data[SUB_REC_POS] = ZERO_BYTE;
  nav_data[SER_RET_POS] = ZERO_BYTE;

 }                  // End of method Nav_Message

//Method update_msg updates nav parameters for class
void update_msg(int[] cmd_update)
 {
  int num_of_128,remainder;
  int num_length;
  int index;
  for (index = 0; index < MSG_LENGTH; index++)
   nav_data[index] = ZERO_BYTE;           // Zero out message array
  nav_data[C_ACL_POS + 3] = (byte)(0x0000007f & cmd_update[2]); // Acceration
  nav_data[C_SPD_POS + 3] = (byte)(0x0000007f & cmd_update[0]); // Speed
  num_of_128 = cmd_update[1] / 128;
  nav_data[C_CRS_POS + 2] = (byte)num_of_128;
  remainder = cmd_update[1] % 128;
  nav_data[C_CRS_POS + 3] = (byte)remainder;
  nav_data[MSG_TYPE_POS] = NAV_TYPE;
  nav_data[REC_TYPE_POS] = NAV_CMD;
  nav_data[SUB_REC_POS] = ZERO_BYTE;
  nav_data[SER_RET_POS] = ZERO_BYTE;
```

```
 }                  // End of update_msg

// Method get_nav_msg retrieves nav message
void get_nav_msg(byte[] init_msg)
 {
  int index;
  for (index = 0; index < MSG_LENGTH; index++)
    init_msg[index] = nav_data[index];
 }                  // End of get init_msg

// Method get_ship_info retrieves ship's position and course
void get_ship_info(byte[] rec_buf,int[] ship_info)
 {
  int index,num_128,rem_128;

  // Code to fill in spd field of nav message
  ship_info[3] = 0;
  ship_info[3] = rec_buf[SPD_POS + 3];
  // Code to fill in crs field of nav message
  ship_info[2] = 0;
  num_128 = rec_buf[CRS_POS + 2];
  num_128 = num_128 * 128;
  rem_128 = rec_buf[CRS_POS + 3];
  ship_info[2] = num_128 + rem_128;

  // Code to fill in y pos field of nav message
  ship_info[1] = 0;
  num_128 = rec_buf[Y_POS + 2];
  num_128 = num_128 * 128;
  rem_128 = rec_buf[Y_POS + 3];
  ship_info[1] = num_128 + rem_128;

  // Code to fill in x pos field of nav message
  ship_info[0] = 0;
  num_128 = rec_buf[X_POS + 2];
  num_128 = num_128 * 128;
  rem_128 = rec_buf[X_POS + 3];
  ship_info[0] = num_128 + rem_128;

 }                  // End of get_ship_info


// Method get_ship_init retrieves ship's initialization info
void get_ship_init(byte[] rec_buf,int[] ship_init_info)
 {
  int index,num_128,rem_128;
  ship_init_info[0] = 0;    // Zero out lat field
  ship_init_info[1] = 0;    // Zero out lon field
  ship_init_info[2] = 0;    // Zero out the course field
  ship_init_info[3] = 0;    // Zero out the speed field

  // Parse out the speed field of nav message
  ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];
```

```
 // Parse out the course field of nav message
 num_128 = rec_buf[CRS_PARM_POS + 2];
 num_128 = num_128 * 128;
 rem_128 = rec_buf[CRS_PARM_POS + 3];
 ship_init_info[2] = num_128 + rem_128;

}                    // End of get_ship_init

// Method get_ship_cmd retrieves ship's command course,speed and acceleration
void get_ship_cmd(byte[] rec_buf,int[] ship_cmd_info)
 {
 int index,num_128,rem_128;
 ship_cmd_info[0] = 0;
 ship_cmd_info[1] = 0;
 ship_cmd_info[2] = 0;

 // Parse out the acceleration  field of nav message
 ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

 // Parse out the speed field of nav message
 ship_cmd_info[1] = rec_buf[C_SPD_POS + 3];

 // Parse out the course field of nav message
 num_128 = rec_buf[C_CRS_POS + 2];
 num_128 = num_128 * 128;
 rem_128 = rec_buf[C_CRS_POS + 3];
 ship_cmd_info[0] = num_128 + rem_128;

}                    // End of get_ship_cmd


// Method build_status_msg parses status message received from server
 void build_status_msg(byte[] nav_data, int[] ship_status_info)
  {
  int index,num_128,rem_128;
  int temp_value;                    /          / Used to prevent zeroing of ship_status_info
  for (index = 0; index < MSG_LENGTH; index++)
   nav_data[index] = ZERO_BYTE;          // Zero out message array

  // Code to fill in spd position field of nav message
  nav_data[SPD_POS + 3] = (byte)(0x0000007f & ship_status_info[3]);

  // Code to fill in crs position field of nav message
  num_128 = ship_status_info[2] / 128;
  rem_128 = ship_status_info[2] % 128;
  nav_data[CRS_POS + 3] = (byte)rem_128;
  nav_data[CRS_POS + 2] = (byte)num_128;

  // Code to fill in y position field of nav message
  num_128 = ship_status_info[1] / 128;
  rem_128 = ship_status_info[1] % 128;
  nav_data[Y_POS + 3] = (byte)rem_128;
  nav_data[Y_POS + 2] = (byte)num_128;
  // Code to fill in x position field of nav message
```

```
    num_128 = ship_status_info[0] / 128;
    rem_128 = ship_status_info[0] % 128;
    nav_data[X_POS + 3] = (byte)rem_128;
    nav_data[X_POS + 2] = (byte)num_128;
    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_CMD;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    nav_data[SER_RET_POS] = ZERO_BYTE;

   }                // End of build_status_msg

}                   // End of class Nav_Message

// Class Net_Connection makes socket conection with server
//  and exchanges messages with server.
class Net_Connection implements Runnable, Smart_Constants
 {
 //VARIABLES for class Nav_Message
 // i_addr            - Internet address of client
 // client_sock       - Socket used by client for exchanging data with server
 // input_data        - Stream used for collecting data from server
 // output_data       - Stream used for outputting data to server
 // client_receiver   - Tread used to decouple receiving and sending data
 // receiver_buffer   - Byte buffer used for receiving data from server
 // new_data          - Variable used to indicate if new data has arrived

 InetAddress i_addr;
 Socket client_sock;
 DataInputStream input_data;
 DataOutputStream output_data;
 Thread client_receiver;
 static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
 static boolean new_data = false;

 // Method Net_Connection is constructor for this class
 public Net_Connection(InetAddress i_addr)
  {
  this.i_addr = i_addr;
  }                 // End of Net_Connection

 // Method send_msg sends messages to server
 public boolean send_msg(byte[] msg_to_server)
  {
  if (client_sock == null)
   {
   try
    {
     client_sock = new Socket(i_addr,SERVER_PORT);
     input_data = new DataInputStream(client_sock.getInputStream());
     output_data = new DataOutputStream(client_sock.getOutputStream());
    }
   catch (Exception e)
    {
     System.out.println("Could not create Socket");
```

```
      System.out.println(e);
    }               // End of try - catch logic
   }                // End of if null
  try
   {
    output_data.write(msg_to_server);
   }
  catch (Exception e)
   {
    System.out.println("Error writing to socket");
    System.out.println(e);
    return false;
   }                // End of try - catch block
  return true;
 }                  // End of send_msg

// Method run executes tread
public void run()
 {
  try
   {
    while (true)
     {
      input_data.read(receiver_buffer); //Client receives data here
      new_data = true;
     }              // End of while
   }
  catch (Exception e)
   {
    System.out.println("Could not receive packet");
    System.out.println(e);
    client_receiver = null;
    return;
   }                // End of try - catch block
 }                  // End of run

// Method stop discontinues running of thread
public void stop()
 {
  if (client_receiver != null)
   {
    client_receiver.stop();
   }
  try
   {
    input_data.close();
    output_data.close();
    client_sock.close();
   }
  catch(IOException e)
   {
    System.out.println("error closing socket");
    System.out.println(e);
   }
```

```
 }                  // End of stop

// Method data_receive determines if new data from server has arrived
public boolean data_receive()
 {
  return new_data;
 }                  / /End of data_receive

// Method clear_data_receive clears new_data variable
public void clear_data_receive()
 {
  new_data = false;
 }                  // End of clear_data_receive

//Method get_message retrieves data sent by server for parsing
public void get_message(byte[] received_data)
 {
  int index;
  for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
   received_data[index] = receiver_buffer[index];
 }                  // End of get_message
}                   // End of class Net_Connection
```

APPENDIX J. RT_SERVER.JAVA FOR TCP

```
// The sever portion of the RT_RCUSC software is made up of four Java classes.
// RT_Sever is the driver of the server position of RT_RCUSC.  This class is
// responsible for instantiation of the Model, Net_S_Connection and Nav_Message
// classes.  RT_Server also makes periodical calls to the Model class for updates of the
// model's attributes.

import java.awt.*;
import java.net.*;
import java.io.*;
import java.lang.*;

public class RT_Server
 {

  public static void main (String args[])
   {
    new RT_RCUSC_Server();
   }                    // End of main
 }                      // End of RT_Server


// RT_RCUSC_Server is where all supporting objects for the server
// Object are instantiated.
class RT_RCUSC_Server implements Smart_Constants
 {
  Net_S_Connection server_con;
  Nav_Message nav_msg;
  Model rt_model;
  static boolean keep_looping = true;
  static boolean start_model = false;
  final static int INIT_FIELDS = 4;             // Number of fields in init message
  final static int STATUS_FIELDS = 4;           // Number of fields in statur fields
  final static int CMD_FIELDS = 3;// Number of fields in command message
  final static int TENTH_SEC = 100;             // Tenth of sec counter
  final static int TWO_SEC = 21;                // Twenty tenth = 2 secs
  int two_sec_counter = 0;
  static byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH];     // Byte buffer for receive message
  static byte[] send_buffer = new byte[MAX_MESSAGE_LENGTH];    // Byte buffer for sending message
  static int[] rt_ruccus_init = new int[INIT_FIELDS];          // Integer buffer for initial info
  static int[] rt_ruccus_cmd = new int[CMD_FIELDS];            // Integer buffer for command info
  static int[] rt_ruccus_status = new int[STATUS_FIELDS];      // Integer buffer for status info
  byte rec_type,sub_rec_type;
  FileOutputStream fsnt3_out;                   // Output stream for command received from client
  DataOutputStream dsnt3_out;                   // Provides methods for writing primitive types
  FileOutputStream fsnt_out;                    // Output stream for ship's position and speed
  DataOutputStream dsnt_out;                    // Provides methods for writing primitive types
  String ser_num,crs_string,spd_string,time;


  RT_RCUSC_Server()
```

```
{
try
 {
  long current_time;
  FileOutputStream fsnt3_out = new FileOutputStream("rec_cmd.txt");
 DataOutputStream dsnt3_out = new DataOutputStream(fsnt3_out);
 FileOutputStream fsnt_out = new FileOutputStream("sent_u.txt");
 DataOutputStream dsnt_out = new DataOutputStream(fsnt_out);
 server_con = new Net_S_Connection();          // Instantiate "network logic class"
 nav_msg = new Nav_Message();                  // Instantiate "Naviation logic class"

 // Loops until a stop message from client is received
 while (keep_looping)
  {
  if (server_con.data_receive())               // Checks if client has sent message
   {
   server_con.clear_data_receive();
   server_con.get_message(rec_buffer);
   rec_type = rec_buffer[MSG_TYPE_POS];
   switch (rec_type)
    {
    case NAV_TYPE:
     {
     sub_rec_type = rec_buffer[REC_TYPE_POS];
     switch (sub_rec_type)
      {
      case NAV_INIT:  //Starts ship's position, heading and speed model
       if (start_model == false)
        {
        try
         {
          start_model = true;
          nav_msg.get_ship_init(rec_buffer,rt_ruccus_init);
          rt_model = new
     Model(rt_ruccus_init[0],rt_ruccus_init[1],rt_ruccus_init[2],rt_ruccus_init[3]);
          ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);
          dsnt3_out.writeBytes(ser_num);
          dsnt3_out.writeBytes("    ");
          crs_string = Integer.toString(rt_ruccus_init[2]);
          dsnt3_out.writeBytes(crs_string);
          spd_string = Integer.toString(rt_ruccus_init[3]);
          dsnt3_out.writeBytes("    ");
          dsnt3_out.writeBytes(spd_string);
          dsnt3_out.writeChar('\n');    //output sent messages to file
         }              // End of try
        catch (IOException ioe)
         {
          System.out.println(" Could not write init to rec_cmd.txt output file");
        }              // End of catch
       }               // End of start_model = false
      break;
     case NAV_CMD:
      nav_msg.get_ship_cmd(rec_buffer,rt_ruccus_cmd); //Parse command
      try
```

```
        {
         ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);
         dsnt3_out.writeBytes(ser_num);
         dsnt3_out.writeBytes("    ");
         crs_string = Integer.toString(rt_ruccus_cmd[0]);
         dsnt3_out.writeBytes(crs_string);
         spd_string = Integer.toString(rt_ruccus_cmd[1]);
         dsnt3_out.writeBytes("    ");
         dsnt3_out.writeBytes(spd_string);
         dsnt3_out.writeChar('\n');        // Output command message received from client
        }                  // End of  try
       catch (IOException ioe)
        {
         System.out.println(" Could not write cmd to rec_cmd.txt output file");
        }                  // End of catch
       rt_model.set_cmd_data(rt_ruccus_cmd);
       break;
      default:// Do nothing with nav stat messages
     }                     // End of switch for sub type
    }                      //End of case NAV_TYPE
    break;
   case MODEL_STOP:
    start_model = false;
    keep_looping = false;
    server_con.stop();             // Stop server_con
    try
     {
      fsnt3_out.close();
      dsnt_out.close();
     }
    catch (IOException ioe)
     {
      System.out.println("could not close file");
     }
    break;
     default:              // Shouldn't be getting her
  } //end of switch for type
 }   // end of if data_receive
try
 {
 Thread.sleep(TENTH_SEC); // Tenth of a second delay
 }
catch (InterruptedException e) { }        // Sleep exception
if (((two_sec_counter % TWO_SEC) == 0) && (start_model == true))
 {
 two_sec_counter = 1;
 rt_model.ship_status(rt_ruccus_status);
 nav_msg.build_status_msg(send_buffer,rt_ruccus_status);
 ser_num = Byte.toString(send_buffer[SER_NUM_LOC]);
 current_time = System.currentTimeMillis();
 time = Long.toString(current_time);
 dsnt_out.writeBytes(ser_num);
 dsnt_out.writeBytes("    ");
 dsnt_out.writeBytes(time);
```

```
      dsnt_out.writeChar('\n');              // Output sent messages to file
      server_con.send_msg(send_buffer);      // Send rt_ruccus update to client
     }                                       // End if 2 second update block
    else
     two_sec_counter++;
   }             // End of while keep_looping
  }              // End of opening files
 catch (IOException ioe)
  {
  System.out.println(" Could not open rec_cmd.txt output file");
  }
 }              // End of RT_RCUSC_Server constructure

}               // End of RT_RCUSC_Server class


/** Class Model updates SmartShip position based on a 2 second interval.
*/
class Model
 {
        // VARIABLES for class Model
        // array_index     - Integer variable used as array index
        // ship_x_pos      - Double variable, x position of SmartShip
        // ship_y_pos      - Double variable, y position of SmartShip
        // ship_crs        - Integer variable, SmartShip course
        // ship_spd        - Integer variable, SmartShip speed
        // ship_lat        - Integer variable, SmartShip latitude
        // ship_lon        - Integer variable, SmartShip longitude
        // ship_cmd_crs    - Integer variable, SmartShip command course
        // ship_cmd_spd    - Integer variable, SmartShip command speed
        // ship_acceration- Integer variable, Smartship acceleration
        // deg_per_2sec    - Constant Integer, maximum number of degs ship can turn
        //                 -    in a position update interval
        // max_delta       - Constant integer, sets heading to cmd heading if
        //                 -    within 5 degs
        // full_compass    - Constant integer, number of degrees in a compass
        // half_compass    - Constant integer, number of degrees in half a compass
        // FEET_MILE       - Constant double, number of foots in mile
        // SECS_HOUR       - Constant double, number of seconds in a hour
        // SECS_UPDATE     - Constant double, number of seconds between update
        // UNITS           - Constant double, number of feet each pixel represents

 static int array_index = 0;
 static double ship_x_pos = 0.0;
 static double ship_y_pos = 0.0;
 static int ship_crs = 0;
 static int ship_spd = 0;
 static int ship_lat = 0;
 static int ship_lon = 0;
 static int ship_cmd_crs = 345;
 static int ship_cmd_spd = 20;
 static int ship_acceration = 1;
 final int deg_per_2sec = 5;
 final int max_delta = 355;
```

```
final int full_compass = 360;
final int half_compass = 180;
final double FEET_MILE = 5280.0;
final double SECS_HOUR = 3600.0;
final double SECS_UPDATE  = 3.0;
final double UNITS = 40.0;  // 7/13 was 10 increase to expand map

/** Model(int,int,int,int) Constructor for class Model - 4 parameters initial latitude, longitude,
 * @param srt_lat is of type int - starting latitude of Smart Ship.
 * @param srt_lon is of type int - starting longitude of Smart Ship.
 * @param srt_crs is of type int - starting course of Smart Ship at begining of interval.
 * @param srt_spd is of type int - starting speed of Smart Ship at begining of interval.
 * @param ship_cmd_spd of type int - command speed of Smart Ship.
 * @param ship_cmd_crs of type int - command course of Smart Ship.
 *   course and speed.
*/

Model(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
 {
 ship_lat = srt_lat;
 ship_lon = srt_lon;
 ship_crs = srt_crs;
 ship_spd = srt_spd;
 ship_cmd_spd = srt_spd;
 ship_cmd_crs = srt_crs;
 }              // End of Model




/** Method ship_status(int[]) updates current SmartShip position, speed and course.
 * @param status array of int current position ,speed and course of Smart Ship.
*/
public boolean ship_status(int status[])
 {
       // VARIABLES for method ship_status
       // delta_crs              - Integer variable, delta between heading and
       //                        -  command heading
       // rad_to_deg             - Double variable, conversion factor for degrees to radians
       // crs_rads               - Double variable, heading in radians
       // temp_x                 - Double variable, used for computation
       // temp_y                 - Double variable, used for computation
 int delta_crs = 0;
 double rad_to_deg = Math.PI / 180.0;
 double crs_rads;
 double temp_x,temp_y;


 if (ship_crs != ship_cmd_crs)
  {
```

```
      delta_crs = Math.abs(ship_crs - ship_cmd_crs);
      if ((delta_crs <= deg_per_2sec) || (delta_crs >= max_delta))
       ship_crs = ship_cmd_crs;
      else
       {
       delta_crs = ship_cmd_crs - ship_crs;
       if ((delta_crs > 0) && (delta_crs <= half_compass))
         ship_crs = ship_crs + deg_per_2sec;
       else if ((delta_crs > 0) && (delta_crs > half_compass))
         ship_crs = ship_crs - deg_per_2sec;
       else if ((delta_crs < 0) && (Math.abs(delta_crs) > half_compass))
         ship_crs = ship_crs + deg_per_2sec;
       else
         ship_crs = ship_crs - deg_per_2sec;
       if (ship_crs > full_compass)
         ship_crs = ship_crs - full_compass;
       if (ship_crs < 0)
        ship_crs = ship_crs + full_compass;
       }              // End of else delta crs !<= deg_per_2sec
     }                // End of cmd_crs != ship_crs
    if (ship_spd != ship_cmd_spd)
     {
     if (ship_spd > ship_cmd_spd)
      ship_spd = ship_spd - ship_acceration;
     else
      ship_spd = ship_spd + ship_acceration;
     }                // End of ship_spd != ship_cmd_spd
    crs_rads = ((double) ship_crs * rad_to_deg);
    temp_x = (((Math.cos(crs_rads) * (double) ship_spd * FEET_MILE) /
          (SECS_HOUR * SECS_UPDATE)) / UNITS);
    temp_y = (((Math.sin(crs_rads) * (double) ship_spd * FEET_MILE) /
          (SECS_HOUR * SECS_UPDATE)) / UNITS);
    ship_x_pos = ship_x_pos + temp_x;
    ship_y_pos = ship_y_pos + temp_y;
    status[0] = (int) Math.round(ship_x_pos);
    status[1] = (int) Math.round(ship_y_pos);
    status[2] = ship_crs;
    status[3] = ship_spd;
    return true;
   }                  // End of ship_status

  /** Method set_cmd_data(int[]) updates model object with  new
   *   command speed and heading.
   * @param cmd_info array of int containing current command data for Smart Ship.
  */
  public boolean set_cmd_data(int cmd_info[])
   {
    ship_cmd_spd = cmd_info[1];
    ship_cmd_crs = cmd_info[0];
    return true;
   }                  // End of set_cmd_data

}                     // End of class model
```

```
// Class Net_S_Connection blocks for socket connection with client,
//  sends and receives data with the client.
class Net_S_Connection implements Runnable, Smart_Constants
 {
 // VARIABLES for class Net_S_Connection
 // client_addr          - URL address for computer running applet
 // server_sock          - Server socket used to establish socket with client
 // sock                 - Socket used for exchanging data with client
 // input_data           - Input stream used to receive data from client
 // output_data          - Output stream used to send data to client
 // server_receiver      - Thread used to decouple receiving and sending data
 // client_port          - Port address blocking for client
 // current_time         - Variable to record current time
 // time                 - String used to output time
 // ser_num              - Serial number of a message
 // receiver_buffer      - Byte buffer to hold input and output messages
 // new_data             - Variable indicating arrival of new data
 // fsn2_out             - File used to output STAT messages from client
 // dsnt2_out            - Used to output STAT messages from client
 static InetAddress client_addr;
 ServerSocket server_sock;
 Socket sock;
 DataInputStream input_data;
 DataOutputStream output_data;
 Thread server_receiver;
 int client_port;
 long current_time;
 String time,ser_num;
 static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
 static boolean new_data = false;
 FileOutputStream fsnt2_out;
 DataOutputStream dsnt2_out;


 // Constructor method for class Net_S_Connection
 public Net_S_Connection()
  {
  if (server_receiver == null)
   {
    server_receiver = new Thread(this);
    server_receiver.start();
   }              // End of server_receiver
  }               // End of Net_S_Connection



// Task that blocks until client establishes socket with server
//  then receives messages from client.
 public void run()
  {
  try
   {
    server_sock = new ServerSocket(SERVER_PORT);
```

```
    sock = server_sock.accept();    // Blocks for a connection
   }
   catch (Exception e)
   {
    System.out.println("Could not create Server_sock");
    System.out.println(e);
   }
   try
   {
    input_data = new DataInputStream(sock.getInputStream());
    output_data = new DataOutputStream(sock.getOutputStream());
    FileOutputStream fsnt2_out = new FileOutputStream("stat_u.txt");
    DataOutputStream dsnt2_out = new DataOutputStream(fsnt2_out);
    try
    {
     while (true)
      {
      try
       {
        input_data.read(receiver_buffer);  //server receives data
       }            // End of try
      catch(IOException e)
       {
        System.out.println("Socket could not receive a packet");
        System.out.println(e);
        System.exit(0);
       }            // End of try catch logic
      if (receiver_buffer[REC_SUB_TYPE_POS] != NAV_STAT)
       {
        new_data = true;              // Might have to implement buffer
       }
      else
       {            // Outputs NAV_STAT message to file
        ser_num = Byte.toString(receiver_buffer[SER_NUM_LOC]);
        current_time = System.currentTimeMillis();
        time = Long.toString(current_time);
        dsnt2_out.writeBytes(ser_num);
        dsnt2_out.writeBytes("    ");
        dsnt2_out.writeBytes(time);
        dsnt2_out.writeChar('\n');    // Output sent messages to file
       }            // Else print out info to file
      }             // End of while
    }    // End of try
    catch (IOException e)
    {
     System.out.println("error opening streams or output files");
     System.out.println(e);
    }               // End of catch for file opening
   }                // End of open file
   catch (IOException ioe)
   {
    System.out.println(" Could not open sent_u.txt output file");
   }                // End of catch
}                   // End of run
```

```
//Method send_msg sends ship's poition and heading
// t o client.
public boolean send_msg(byte[] msg_to_server)
 {
 try
  {
    output_data.write(msg_to_server);
  }      // End of try
 catch (Exception e)
  {
   System.out.println("Error sending data");
   System.out.println(e);
   return false;
  }      // End of try - catch block
 return true;
}       // End of send_msg

//Method to unplug socket
public void stop()
 {
 if (server_receiver != null)
  {
   server_receiver.stop();
  }
 try
  {
   input_data.close();
   output_data.close();
   server_sock.close();
  }               // End of try
 catch(IOException e)
  {
   System.out.println("error closing socket");
   System.out.println(e);
  }               // End of catch
 try
  {
   fsnt2_out.close();
  }               // End of try
 catch (IOException ioe)
  {
   System.out.println("could not close file");
  }               // End of catch

 }               // End of stop

//Method data_receive indicates if new data has been received.
public boolean data_receive()
 {
  return new_data;
 }               // End of data_receive

//Method clear_data clears new data variable
```

```
 public void clear_data_receive()
  {
   new_data = false;
  }                  // End of clear_data_receive


 //Method get_message retrives data received from client for
 //  processing.
 public void get_message(byte[] received_data)
  {
   int index;
   for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
    received_data[index] = receiver_buffer[index];
  }                  // End of get_message

}                    // End of class Net_S_Connection


// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants
 {
 // VARIABLES for class Nav_Message
 // MSG_LENGTH              - Max length for nav message is 36 bytes
 // NUM_OF_SHIP_PARMS       - Nav message contains 4 parameters for ship
 // LAT_PARM_POS            - Latitude parameter position in byte buffer
 // LON_PARM_POS            - Longitude parameter position in byte buffer
 // CRS_PARM_POS            - Course parameter position in byte buffer
 // SPD_PARM_POS            - Speed parameter position in byte buffer
 // Y_POS                   - Ship's y position in byte buffer
 // X_POS                   - Ship's x position in byte buffer
 // CRS_POS                 - Ship's course position in byte buffer
 // SPD_POS                 - Ship's speed position in byte buffer
 // C_CRS_POS               - Ship's command course position in byte buffer
 // C_SPD_POS               - Ship's command speed position in byte buffer
 // C_ACL_POS               - Ship's command acceleration in byte buffer
 // PARM_LENGTH             - Parameter's length - 4 bytes
 // ZERO_BYTE               - Byte of 0's
 // index2                  - Used for loop variable
 // ser_number              - Serial number of message
 // MAX_SERIAL              - Largest serial number after which value is set to 0
 // nav_data                - Byte buffer used for storing messages
 // char_digit              - Character used to store a digit as a character
 final static int MSG_LENGTH = 36;
 final static int NUM_OF_SHIP_PARMS = 4;   // x pos,y pos,course and speed
 final static int LAT_PARM_POS = 4;
 final static int LON_PARM_POS = 8;
 final static int CRS_PARM_POS = 12;
 final static int SPD_PARM_POS = 16;
 final static int X_POS = 4;
 final static int Y_POS = 8;
 final static int CRS_POS = 12;
 final static int SPD_POS = 16;
 final static int C_CRS_POS = 4;
```

```
 final static int C_SPD_POS = 8;
 final static int C_ACL_POS = 12;
 final static int PARM_LENGTH = 4;
 final static byte ZERO_BYTE = 0;
 int index2;
 static byte ser_number = 0;
 static final byte MAX_SERIAL= 125;
 static byte[] nav_data = new byte[MSG_LENGTH];
 char char_digit;

 /** Nav_Message Constructor for class
 */
 Nav_Message()
  {

  }                  // Constructor with no arguments


 /** Nav_Message(int,int,int,int) Constructor for class - 4 parameters
  * @param srt_lat is starting latitude of Smart Ship.
  * @param srt_lon is starting longitude of Smart Ship.
  * @param srt_crs is starting course of Smart Ship.
  * @param srt_spd is starting speed of Smart Ship.
 */
 Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
  {
   int index,index2;
   int num_length,num_128,rem_128;

   for (index = 0; index < MSG_LENGTH; index++)
     nav_data[index] = ZERO_BYTE;          // Zero out message array

   // Code to fill in speed field of nav message
   nav_data[SPD_PARM_POS + 3] = (byte)(0x0000007f & srt_spd);

   // Code to fill in course field of nav message
   num_128 = srt_crs / 128;
   rem_128 = srt_crs % 128;
   nav_data[CRS_PARM_POS + 3] = (byte)rem_128;
   nav_data[CRS_PARM_POS + 2] = (byte)num_128;
   nav_data[MSG_TYPE_POS] = NAV_TYPE;
   nav_data[REC_TYPE_POS] = NAV_INIT;
   nav_data[SUB_REC_POS] = ZERO_BYTE;
   nav_data[SER_RET_POS] = ZERO_BYTE;
  }                  // End of method Nav_Message

 // Method update_msg updates nav parameters for class
 void update_msg(int[] cmd_update)
  {
   int index,index2,num_128,rem_128;
   int num_length;

   for (index = 0; index < MSG_LENGTH; index++)
    nav_data[index] = ZERO_BYTE;          // Zero out message array
```

```
   // Code to fill in acceleration field of nav message
   nav_data[C_ACL_POS + 3] = (byte)(0x0000007f & cmd_update[2]);

   // Code to fill in speed field on nav message
   nav_data[C_SPD_POS + 3] = (byte)(0x0000007f & cmd_update[1]);

   // Code to fill in crs field of nav message
   num_128 = cmd_update[0] / 128;
   rem_128 = cmd_update[0] % 128;
   nav_data[C_CRS_POS + 3] = (byte)rem_128;
   nav_data[C_CRS_POS + 2] = (byte)num_128;

   nav_data[MSG_TYPE_POS] = NAV_TYPE;
   nav_data[REC_TYPE_POS] = NAV_CMD;
   nav_data[SUB_REC_POS] = ZERO_BYTE;
   nav_data[SER_RET_POS] = ZERO_BYTE;


  }                // End of update_msg

 //Method get_nav_msg retrieves ship's current data
 void get_nav_msg(byte[] init_msg)
  {
   int index;
   for (index = 0; index < MSG_LENGTH; index++)
    init_msg[index] = nav_data[index];
  }                // End of get init_msg

// Method get_ship_init retrieves ship's initial data
 void get_ship_init(byte[] rec_buf,int[] ship_init_info)
  {
   int index,num_128,rem_128;
   ship_init_info[0] = 0;     // Zero out lat field
   ship_init_info[1] = 0;     // Zero out lon field
   ship_init_info[2] = 0;     // Zero out the course field
   ship_init_info[3] = 0;     // Zero out the speed field

   // Parse out the speed field of nav message
   ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];

   // Parse out the course field of nav message
   num_128 = rec_buf[CRS_PARM_POS + 2];
   num_128 = num_128 * 128;
   rem_128 = rec_buf[CRS_PARM_POS + 3];
   ship_init_info[2] = num_128 + rem_128;

  }                // End of get_ship_init

//Method get_ship_cmd  retrieves ship's command data
void get_ship_cmd(byte[] rec_buf,int[] ship_cmd_info)
 {
  int index,num_128,rem_128;
  ship_cmd_info[0] = 0;
```

```
 ship_cmd_info[1] = 0;
 ship_cmd_info[2] = 0;

 // Parse out the acceleration  field of nav message
 ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

 // Parse out the speed field of nav message
 ship_cmd_info[1] = rec_buf[C_CRS_POS + 3];

 // Parse out the course field of nav message
 num_128 = rec_buf[C_SPD_POS + 2];
 num_128 = num_128 * 128;
 rem_128 = rec_buf[C_SPD_POS + 3];
 ship_cmd_info[0] = num_128 + rem_128;

}                    // End of get_ship_cmd


// Method build_status_msg builds status message to be sent to client
void build_status_msg(byte[] nav_data, int[] ship_status_info)
 {
  int index;
  byte num_128,rem_128;
  for (index = 0; index < MSG_LENGTH; index++)
    nav_data[index] = ZERO_BYTE;         // Zero out message array

 // Code to fill in spd position field of nav message
 nav_data[SPD_POS + 3] = (byte)(0x0000007f & ship_status_info[3]);

 // Code to fill in crs position field of nav message
 num_128 = (byte)(ship_status_info[2] / 128);
 rem_128 = (byte)(ship_status_info[2] % 128);
 nav_data[CRS_POS + 2] = num_128;
 nav_data[CRS_POS + 3] = rem_128;

 // Code to fill in y position field of nav message
 num_128 = (byte)(ship_status_info[1] / 128);
 rem_128 = (byte)(ship_status_info[1] % 128);
 nav_data[Y_POS + 2] = num_128;
 nav_data[Y_POS + 3] = rem_128;

 // Code to fill in x position field of nav message
 num_128 = (byte)(ship_status_info[0] / 128);
 rem_128 = (byte)(ship_status_info[0] % 128);
 nav_data[X_POS + 2] = num_128;
 nav_data[X_POS + 3] = rem_128;

 nav_data[MSG_TYPE_POS] = NAV_TYPE;
 nav_data[REC_TYPE_POS] = NAV_CMD;
 nav_data[SUB_REC_POS] = ZERO_BYTE;
 // Fill in serial number
 nav_data[SER_NUM_LOC] = ser_number;
 ser_number++;
 if (ser_number > MAX_SERIAL)
```

```
    ser_number = 0;
  }                // End of build_status_msg
}                  // End of class Nav_Message
```

APPENDIX K. SMART_CONSTANTS.JAVA FOR TCP

```
public interface Smart_Constants
 {
  final static int BITS_IN_BYTE = 8;          // Number of bits in a byte
  final static int BYTES_IN_INT = 4;          // Number of bytes in a integer
  final static int SER_RET_POS = 3;           // Serial number position in buffer
  final static int MSG_TYPE_POS = 0;          // Message type position in buffer
  final static int REC_TYPE_POS = 1;          // Record type position in buffer
  final static int SUB_REC_POS = 2;           // Sub record type position in buffer
  final static int MAX_MESSAGE_LENGTH = 72; //Maximum bytes in buffer
  final static int MAX_ARRAY = 25;
  final static int SERVER_PORT = 1997;        // Serial port used for sockets
  final static int REC_SUB_TYPE_POS = 1; // Sub record type position in buffer
  final static byte NAV_TYPE = 1;             // NAV_TYPE value
  final static byte NAV_INIT = 1;             // NAV_INIT value
  final static byte NAV_CMD  = 2;             // NAV_CMD value
  final static byte NAV_STAT = 3;             // NAV_STAT value
  final static byte SER_NUM_LOC = 3;          // SER_NUM_LOC type position in buffer
  final static byte MODEL_STOP = 99;          // MODEL_STOP value
}                    // End of Smart_Constants
```

APPENDIX L. RT_RCUSCHPAGE.JAVA FOR UDP

```
// RT_RCUSCHpage.java is the client portion of the RT_RCUSC software.  It is made
// up of seven classes.  RT_RCUSCHpage is the driver of the client program and is
// responsible for instantiation of the Navcontrol, Net_Connection. Nav_message
// NavUpdate, Compass and Plot classes.

import java.awt.*;
import java.net.*;
import java.io.*;

// Class RT_RCUSCHpage - Main class of this applet.
//  It's responsible for instantiation of the four
//  supporting objects - NavControl, NavUpdate, Compass,
//  and Plot.
/** RT_RCUSCHpage.java
* @version 1.1
* @author Floyd Bailey
* Implementation for RT_RCUSCHpage main control
*/

/**
 * Constructor for class RT_RCUSCHpage.
 */

public class RT_RCUSCHpage extends java.applet.Applet
 implements Runnable, Smart_Constants
 {
        // VARIABLES for RT_RCUSCHpage
        // init_nav             - NavControl object declaration
        // update_nav           - NavUpdate object declaration
        // compass_update       - Compass object declaration
        // plot_update          - Plot object declaration
        // ship_model           - Model object declaration
        // first_time           - Boolean variable indicating whether
        //                      -   initial course and speed was entered
        // i                    - Integer variable used for exception
        //                      - handler
        // InitThread           - Thread object declaration
        // MAX_ARRAY            - Integer constant used to determine the
        //                      - size of array
        // point_history        - Array of Point used to be last MAX_ARRAY
        //                      - x,y position of RT_RCUSC
        // temp_point           - Temporary Point variable
        // array_index          - Integer variable indicating the number of
        //                      - Elements in point_history
        // ship_info            - Integer array used to update RT_RCUSC
        //                      - Position parameters
        // cmd_info             - Integer array used to update RT_RCUSC
        //                      - Command parameters
        // ret_value            - Boolean variable assign returning condition
        //                      -   of methods returning boolean type
        // panel1               - Instantiation of Java Panel class
```

```
  NavControl init_nav;
  NavUpdate update_nav;
  Compass compass_update;
  Plot plot_update;
  Nav_Message nav_msg;          // Class to handle building and parsing nav
  Net_Connection net_con;       // Class to handle network connection
  static boolean first_time = true;

  int i = 0;
  Thread InitThread;
  static Point[] point_history = new Point[MAX_ARRAY];
  Point temp_point,temp_point2;
  static int array_index = 0;
  static int[] ship_info = new int[4];
  static int[] cmd_info = new int[3];
  static int update_trigger = 0;      // Used to trigger update every 2 sec
  boolean ret_value;
  String host;
  static byte[] msg_buffer = new byte[MAX_MESSAGE_LENGTH];
  byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH];
  Panel panel1 = new Panel();
  byte record_type;




/** Method init() of class RT_RCUSC - This method instantiates objects
 *  of classes NavUpdate, Compass, Plot and NavUpdate.
*/
  public void init()
   {
   host = this.getCodeBase().getHost();
   init_nav = new NavControl(this, "LAT", "LON", "X_POS", "Y_POS", "COURSE", "SPEED",
                             0, 0, 0, 0, 0, 0);
   update_nav = new NavUpdate(this, "COURSE", "SPEED", host, 0, 0, 0);
   compass_update = new Compass(this);
   plot_update = new Plot(this);
   setBackground(Color.white);
   setLayout(new GridLayout(2,2,10,10));
   add(init_nav);
   add(update_nav);
   add(compass_update);
   add(plot_update);
   // Get address of applet's home
   }              // End of init

/** Method insets(int,int,int,int) of class RT_RCUSC - This method
 *  determines the spaces between the four panels - init_nav,
 *  update_nav, compass_update and plot_update.
*/

  public Insets insets()
```

```
  {
   return new Insets(10,10,10,10);
  }                    // End of insets

/** Method start() of class RT_RCUSC - This method instantiates thread
 *  InitThread and starts it.
*/
 public void start()
  {
   if (InitThread == null)
    {
     InitThread = new Thread(this);
     InitThread.start();
    }                          // End of == null
  }                            // End of start

/** Method stop() of class RT_RCUSC - this method stops thread
 *  InitThread and de-instantiates it.
*/
 public void stop()
  {
   if (InitThread != null)
    {
     InitThread.stop();                // Put back in 5/25
     InitThread = null;
     net_con.stop();                   // Disconnect connection
    }                                  // End of != null
  }                                    // End of stop




/** Method run() of class RT_RCUSC - This method is the main loop of the
 *  class.  This is where data is exchanged between the five supporting
 *  objects - init_nav, update_nav, compass_update, plot_update and
 *  ship_model.
*/
 public void run()
  {
   while (true)                // Loop forever
    {
     if (init_nav.running())
      {
       if (first_time)           // Get RT_RCUSC initial speed and course
        {
        try
         {
          ret_value = init_nav.init_info(ship_info);
          nav_msg = new Nav_Message(ship_info[0],ship_info[1],ship_info[2],ship_info[3]);
          nav_msg.get_nav_msg(msg_buffer);          // Get msg to send
          net_con = new Net_Connection(InetAddress.getByName(host));
          net_con.send_msg(msg_buffer);     // Send out init position to server
```

```
    first_time = false;
  }                    // End of try
 catch (Exception e)
  {
   System.out.println(e);
  }          // End of catch
}                      // End of first_time
else                   // RT_RCUSC initial data already entered
{
 if (update_trigger >= 23)  // TICKLE VALUE - change to another number ex 27 & recompile
  {                    // Get RT_RCUSC command speed, course and acceleration update
   update_trigger = 0;
   cmd_info[0] = 25;           // Speed
   cmd_info[1] = cmd_info[1] + 3;     // Course
   if (cmd_info[1] > 359)
    cmd_info[1] = 0;
   nav_msg.update_msg(cmd_info);      // Nav_msg class builds update msg
   nav_msg.get_nav_msg(msg_buffer);  // Get msg to send
   net_con.send_msg(msg_buffer);          //Send msg to server
  }                    // End of if update_command
 if (net_con.data_receive())              // Has server sent info
  {
   net_con.clear_data_receive();
   net_con.get_message(rec_buffer);
  record_type = rec_buffer[0];
  switch (record_type)                    // Parse server info
   {
   case NAV_TYPE :
    nav_msg.get_ship_info(rec_buffer,ship_info);
    rec_buffer[REC_SUB_TYPE_POS] = NAV_STAT;
    net_con.send_msg(rec_buffer);    //Send msg back to server for timing
    break;
   default:                                // Shouldn't be getting here
   }                                       // End of switch logic
   ret_value = compass_update.set_compass(ship_info[2]);
 }                                         // End of if data_receive
 if (array_index < MAX_ARRAY)
 {
  temp_point = new Point(ship_info[0],ship_info[1]);
  point_history[array_index] = temp_point;
  array_index++;
 }                                         // End of if array not full
else
 {
  temp_point2 = point_history[MAX_ARRAY - 1];
  for (int for_index = (MAX_ARRAY - 1); for_index > 0; for_index--)
   {
    temp_point = point_history[for_index - 1];
    point_history[for_index - 1] = temp_point2;
    temp_point2 = temp_point;
   }                                       // End of for loop
  temp_point = new Point(ship_info[0],ship_info[1]);
  point_history[MAX_ARRAY - 1] = temp_point;
 }                                         // End of else array not full yet
```

```
        ret_value = plot_update.plot_array(point_history,array_index);
        repaint();
        compass_update.repaint();
        plot_update.repaint();
       }                                        // End of else
      try
       {
        Thread.sleep(100); // one tenth of second delay between positon updates
        update_trigger++;
       }
      catch (InterruptedException e) { }        // Sleep exception
     }                                          // End of if running
    repaint();
   }                                            // End of while (true) loop
  }                                             // End of run

/** Method update(NavControl) of class RT_RCUSC - TBD */

void update(NavControl Nav_in)
  {
        //Future code can go here
  }                             // End of update NavControl

 /** Method update(NavUpdate) of class RT_RCUSC - TBD */
 void update(NavUpdate Nav_in)
  {
        // Future code can go here
  }                             // End of update NavUpdate

 /** Method paint(Graphics)   */
 public void paint(Graphics g)
  {
   g.setColor(Color.red);
   g.drawString("       RUCUS ",0, 60);
  }                             // End of pairing
 }

/** Class NavControl  - Supporting class of RT_RCUSCHpage.
 *  It's responsible for initial RT_RCUSC data
*/
class NavControl extends Panel
 {
        // VARIABLES for class NavControl
        // running          - Boolean variable which indicates if object is
        //                  -    running
        // lat              - TextField variable to output latitude data to screen
        // lon              - TextField variable to output longitude to screen
        // x_pos            - TextField variable to output x position to screen
        // y_pos            - TextField variable to output y position to screen
        // course           - TextField variable to output course to screen
        // speed            - TextField variable to output speed to screen
        // r_lat            - Integer variable for current latitude of RT_RCUSC
        // r_lon            - Integer variable for current longitude of RT_RCUSC
        // r_x_pos          - Integer variable for current x position of RT_RCUSC
```

```
// r_y_pos        - Integer variable for current y position of RT_RCUSC
// r_icourse      - Integer variable for current RT_RCUSC heading
// r_ispeed       - Integer variable for current RT_RCUSC speed
// MAX_SPEED      - Constant maximum speed of RT_RCUSC
// MIN_SPEED      - Constant minimum speed of RT_RCUSC
// MAX_CRS        - Constant maximum allowable compass input
// MIN_CRS        - Constant minimum allowable compass input
// MIN_X          - Constant minimum allowable x axis value
// MAX_X          - Constant maximum allowable x axis value
// MIN_Y          - Constant minimum allowable y axis value
// MAX_Y          - Constant maximum allowable y axis value
// MAX_LAT        - Constant maximum allowable latitude value
// MIN_LAT        - Constant minimum allowable latitude value
// MAX_LON        - Constant maximum allowable longitude value
// MIN_LON        - Constant minimum allowable longitude value
// start_button   - Constant String value for push button label
// stop_button    - Constant String value for pust button label
// outerparent    - Variable used to declare RT_RCUSCHpage as parent
//                -    to NavControl

static boolean running = false;
static TextField lat,lon,x_pos,y_pos,course,speed;
static int r_lat = 0;
static int r_lon = 0;
static int r_x_pos = 0;
static int r_y_pos = 0;
static int r_icourse = 0;
static int r_ispeed = 0;
final int MAX_SPD = 45;
final int MIN_SPD = -20;
final int MAX_CRS = 360;
final int MIN_CRS = 0;
final int MIN_X = -1024;
final int MAX_X = 1024;
final int MAX_Y = 1024;
final int MIN_Y = -1024;
final int MAX_LAT = 90;
final int MIN_LAT = -90;
final int MIN_LON = -180;
final int MAX_LON = 180;
final String start_button = "START";
final String stop_button = "STOP";
RT_RCUSCHpage outerparent;

/** NavControl  is the Constructor for class NavControl.  It takes 12
 *  parameters 6 of type String and 6 parameters of double.
* @param parm1 string label for latitude of RT_RCUSC.
* @param parm2 string label for longitude of RT_RCUSC.
* @param parm3 string label for x position of RT_RCUSC.
* @param parm4 string label for y position of RT_RCUSC.
 * @param parm5 string label for course of RT_RCUSC.
* @param parm6 string label for speed of RT_RCUSC.
* @param flt1 initial Latitude of RT_RCUSC.
* @param flt2 initial Longitude of RT_RCUSC.
```

```
        * @param flt3 initial x position of RT_RCUSC 0.
        * @param flt4 initial y position of RT_RCUSC 0.
        * @param flt5 initial course of RT_RCUSC.
    * @param flt6 initial speed of RT_RCUSC.
    */
NavControl(RT_RCUSCHpage target,String parm1, String parm2, String parm3,
        String parm4, String parm5, String parm6, double flt1, double flt2,
        double flt3, double flt4, double flt5, double flt6)
{
 this.outerparent = target;
 setLayout(new GridLayout(7,2,10,10));  // Sets up a grid of 7 rows and 2 columns

 // Convert float parameters to string
 lat = new TextField(String.valueOf(flt1),6);
 lon = new TextField(String.valueOf(flt2),6);
 x_pos = new TextField(String.valueOf(flt3),6);
 y_pos = new TextField(String.valueOf(flt4),6);
 course = new TextField(String.valueOf(flt5),6);
 speed = new TextField(String.valueOf(flt6),6);



 // Set edit boxes to a white background
 lat.setBackground(Color.white);
 lon.setBackground(Color.white);
 x_pos.setBackground(Color.white);
 y_pos.setBackground(Color.white);
 course.setBackground(Color.white);
 speed.setBackground(Color.white);

 // Create label and text fields
 add(new Label(parm1, Label.RIGHT));
 add(lat);
 add(new Label(parm2, Label.RIGHT));
 add(lon);
 add(new Label(parm3, Label.RIGHT));
 add(x_pos);
 add(new Label(parm4, Label.RIGHT));
 add(y_pos);
 add(new Label(parm5, Label.RIGHT));
 add(course);
 add(new Label(parm6, Label.RIGHT));
 add(speed);

 // Create and label two buttons
 add(new Button("START"));
 add(new Button("STOP"));
 setBackground(Color.green);
}               // End of constructor NavControl

/** Method insets sets up the spacing between items in this panel - labels, text boxes
 *    and buttons.
*/
```

```
public Insets insets()
 {
 return new Insets(10,10,10,150);
 }                          // End of insets

/** Method init_info loads initial latitude, longitude, course and
 *   speed into an array.
*/
public boolean init_info(int info[])
 {
  info[0] = r_lat;
  info[1] = r_lon;
  info[2] = r_icourse;
  info[3] = r_ispeed;
  return true;
 }                          // End of init_info
```

```
/** Method action(Event,Object) handles selection of buttons
 *   and TextFields on this panel.
 * @param evt type of event to trigger this action
 * @param arg object type causing this action
*/
 public boolean action(Event evt, Object arg)
  {
  if (evt.target instanceof TextField)
   {
         outerparent.update(this);
         return true;
   }                        // End of if true
  else if (evt.target instanceof Button)
   {
   String label = (String)arg;
   if (label.equals(start_button))
    {
     read_values();
     return true;
    }                       // End of if START button was selected
   else
    {
     running = false;
     return true;
    }                       //  STOP button was selected.
  }
 else
  return false;
 }                          // End of action
```

```
/** Method running returns a boolean type indicating what
 *     state this panel is in - running or not running.
*/
public boolean running()
 {
  return running;
 }                          // End of running

 /** Method read_values reads and checks for values entered by the user.
 */
 public void read_values()
 {
        // VARIABLES for method read_values
        // crs_string      - String variable used to store course data
        // spd_string      - String variable used to store speed data
        // xpos_string     - String variable used to store x position data
        // ypos_string     - String variable used to store y position data
        // lat_string      - String variable used to store latitude data
        // lon_string      - String variable used to store longitude data
        // crs_good        - Boolean variable indicating proper course entry
        // spd_good        - Boolean variable indicating proper speed entry
        // xpos_good       - Boolean variable indicating proper x position entry
        // ypos_good       - Boolean variable indicating proper y position entry
        // lat_good        - Boolean variable indicating proper latitude entry
        // lon_good        - Boolean variable indicating proper longitude entry
        // tmp_spd         - Integer variable used for computations
        // tmp_crs         - Integer variable used for computations
        // tmp_xpos        - Integer variable used for computations
        // tmp_ypos        - Integer variable used for computations
        // tmp_lat         - Integer variable used for computations
        // tmp_lon         - Integer variable used for computations
 String crs_string;
 String spd_string;
 String xpos_string;
 String ypos_string;
 String lat_string;
 String lon_string;
 boolean crs_good;
 boolean spd_good;
 boolean xpos_good;
 boolean ypos_good;
 boolean lat_good;
 boolean lon_good;
 int tmp_spd = r_ispeed;
 int tmp_crs = r_icourse;
 int tmp_xpos = r_x_pos;
 int tmp_ypos = r_y_pos;
 int tmp_lat = r_lat;
 int tmp_lon = r_lon;

 // Next 7 instructions assigns values enter by user into String variables
 crs_string = course.getText();
 spd_string = speed.getText();
```

```
xpos_string = x_pos.getText();
ypos_string = y_pos.getText();
lat_string = lat.getText();
lon_string = lon.getText();

// Initialize data as valid.
crs_good = spd_good = xpos_good = ypos_good = lat_good = lon_good = true;

// Following try block parses user data and checks for correct range and type.
try
 {
  tmp_crs = Integer.parseInt(crs_string);
  if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
   {
    crs_good = false;
    course.setBackground(Color.red);
    course.setText(String.valueOf(r_icourse));
   }            // End of failed course entry
 tmp_spd = Integer.parseInt(spd_string);
 if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
  {
   spd_good = false;
   speed.setBackground(Color.red);
   speed.setText(String.valueOf(r_ispeed));
  }             // End of failed speed entry
 tmp_xpos = Integer.parseInt(xpos_string);
 if ((tmp_xpos > MAX_X) || (tmp_xpos < MIN_X))
  {
   xpos_good = false;
   x_pos.setBackground(Color.red);
   x_pos.setText(String.valueOf(r_x_pos));
  }             // End of failed x pos entry
 tmp_ypos = Integer.parseInt(ypos_string);
 if ((tmp_ypos > MAX_Y) || (tmp_ypos < MIN_Y))
  {
   xpos_good = false;
   y_pos.setBackground(Color.red);
   y_pos.setText(String.valueOf(r_y_pos));
  }             // End of failed y pos entry
 tmp_lat = Integer.parseInt(lat_string);
 if ((tmp_lat > MAX_LAT) || (tmp_lat < MIN_LAT))
  {
   lat_good = false;
   lat.setBackground(Color.red);
   lat.setText(String.valueOf(r_lat));
  }             // End of failed lat entry
 tmp_lon = Integer.parseInt(lon_string);
 if ((tmp_lon > MAX_LON) || (tmp_lon < MIN_LON))
  {
   lon_good = false;
   lon.setBackground(Color.red);
   lon.setText(String.valueOf(r_lon));
  }             // End of failed lon entry
}               // End of try
```

```
    catch (Exception e)
     {

     }              // End of catch
    if ((crs_good == true) && (spd_good == true) && (xpos_good == true) &&
       (ypos_good == true) && (lat_good == true) && (lon_good == true))
     {
      course.setBackground(Color.white);
      speed.setBackground(Color.white);
      x_pos.setBackground(Color.white);
      y_pos.setBackground(Color.white);
      lat.setBackground(Color.white);
      lon.setBackground(Color.white);
      course.setText(String.valueOf(tmp_crs));
      r_icourse = tmp_crs;
      speed.setText(String.valueOf(tmp_spd));
      r_ispeed = tmp_spd;
      x_pos.setText(String.valueOf(tmp_xpos));
      r_x_pos = tmp_xpos;
      y_pos.setText(String.valueOf(tmp_ypos));
      r_y_pos = tmp_ypos;
      lat.setText(String.valueOf(tmp_lat));
      r_lat = tmp_lat;
      lon.setText(String.valueOf(tmp_lon));
      r_lon = tmp_lon;
      running = true;
     }              // End of good values submitted
   }                // End of read_values
 }                  // End of class NavControl



/** Class NavUpdate is where command data is enter by the user
*/
 class NavUpdate extends Panel
 {
        // VARIABLES for class NavUpdate
        // course          -  String variable used to store course data
        // speed           -  String variable used to store speed data
        // acceleration    -  String variable used to store acceleration data
        // update_cmd      -  Boolean variable indicating if command data was
        //                 -     updated
        // outerparent     -  Object variable indicating that RT_RCUSCHpage is
        //                 -     parent to this class
        // r_spd           -  Integer variable indicating current command speed
        // r_crs           -  Integer variable indicating current command course
        // r_acl           -  Integer variable indicating current command acceleration
        // MAX_SPD         -  Constant integer, maximum speed for RT_RCUSC
        // MIN_SPD         -  Constant integer, minimum speed for RT_RCUSC
        // MAX_CRS         -  Constant integer, maximum course (degrees ) for course
        // MIN_CRS         -  Constant integer, minimum course (degrees ) for course
        // MAX_ACL         -  Constant integer, maximum acceleration for RT_RCUSC
```

```
        // MIN_ACL     -  Constant integer, minimum acceleration for RT_RCUSC
static TextField course,speed,acceleration;
static boolean update_cmd = false;
RT_RCUSCHpage outerparent;
static int r_spd = 0;
static int r_crs = 0;
static int r_acl = 0;
final static int MAX_SPD = 45;
final static int MIN_SPD = -20;
final static int MAX_CRS = 360;
final static int MIN_CRS = 0;
final static int MAX_ACL = 6;
final static int MIN_ACL = -6;

/** Constructor for class NavUpdate - takes 6 parameters.  3 of type String and
 *     3 of type Integer.
 * @param target
 * @param parm1 string type used as a label for Course text field.
 * @param parm2 string type used as a label for Speed text field.
 * @param parm3 string type used as a label for Acceleration text field.
 * @param flt1 integer type used for course enter by user.
 * @param flt2 integer type used for speed enter by user.
 * @param flt3 integer type used for acceleration enter by user.
*/
NavUpdate(RT_RCUSCHpage target,String parm1, String parm2, String parm3,
          int flt1, int flt2, int flt3)
{
 this.outerparent = target;
 setBackground(Color.blue);
 // Next instruction sets up a grid for this panel - 4 rows by 2 colums.
 setLayout(new GridLayout(4,2,10,40));
 String blank = "      "; // Used to align submit button

 // Convert float parameters to string
 course = new TextField(String.valueOf(flt1),5);
 speed = new TextField(String.valueOf(flt2),5);
 acceleration = new TextField(String.valueOf(flt3),5);

 // Set TextField background to white.
 course.setBackground(Color.white);
 speed.setBackground(Color.white);
 acceleration.setBackground(Color.white);

 // Create label and text fields
 add(new Label(parm1, Label.RIGHT));
 add(course);
 add(new Label(parm2, Label.RIGHT));
 add(speed);
 add(new Label(parm3, Label.RIGHT));
 add(acceleration);
 add(new Label(blank, Label.RIGHT));
 add(new Button("SUBMIT"));
}               // End of constructor NavUpdate
```

```
/** Method insets determines spaces between item in this panel - TextFields, labels
*    and button.
*/
public Insets insets()
 {
  return new Insets(10,10,10,70);
 }                          // End of insets

/** This method get_cmd_data(int cmd_info[]) the current command data.
 * @param cmd_info[] is an integer array containing command data
 *   The values are read into an array which the
 *   caller can use to update RT_RCUSC position.
*/
public boolean get_cmd_data(int cmd_info[])
 {
  cmd_info[0] = r_spd;
  cmd_info[1] = r_crs;
  cmd_info[2] = r_acl;
  return true;
 }                          // End of get_cmd_data

/** Method action handles TextField and button action of this panel.
 * @param evt is of type Event, it's the event of this action.
 * @param arg is of type Object, it's the object of this action.
*/
public boolean action(Event evt, Object arg)
 {
  if (evt.target instanceof TextField)
   {
    outerparent.update(this);
    return true;
   }              // End of if true
  else if (evt.target instanceof Button)
   {
    read_values();
    update_cmd = true;
    return true;
   }              // End of else if Button
  else
   return false;
 }              // End of action

/** Method update_command() returns a boolean indicating if command
 *    data has been updated.
*/
public boolean update_command()
 {
  return update_cmd;
 }              // End of update_command


/** Method clear_update_cmd() set update_cmd to false.
*/
public void clear_update_cmd()
```

```
{
 update_cmd = false;
}               // End of clear_update

/** Method read_values() validates and assigns command data entered by user.
*/
public void read_values()
{
        // VARIABLES for method read_values()
        // crs_string      -   String variable used to store course data
        // spd_string      -   String variable used to store speed data
        // acl_string      -   String variable used to store acceleration data
        // crs_good        -   Boolean variable indicating valid course entry
        // spd_good        -   Boolean variable indicating valid speed entry
        // acl_good        -   Boolean variable indicating valid acceleration
        // tmp_spd         -   Integer used for computation
        // tmp_crs         -   Integer used for computation
        // tmp_acl         -   Integer used for computation
String crs_string;
String spd_string;
String acl_string;
boolean crs_good;
boolean spd_good;
boolean acl_good;
int tmp_spd;
int tmp_crs;
int tmp_acl;

// Next 3 instruction assigns command data to String variables.
crs_string = course.getText();              // Get text of course text box
spd_string = speed.getText();               // Get text of speed text box
acl_string = acceleration.getText();        // Get text of acceleration text box

tmp_spd = r_spd;
tmp_crs = r_crs;
tmp_acl = r_acl;

//Initialize valid data to true.
crs_good = spd_good = acl_good = true;
try
{
 tmp_crs = Integer.parseInt(crs_string);
 if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
  {
   crs_good = false;
   course.setBackground(Color.red);
   course.setText(String.valueOf(r_crs));
  }             // End of failed course entry
 tmp_spd = Integer.parseInt(spd_string);
 if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
  {
   spd_good = false;
   speed.setBackground(Color.red);
   speed.setText(String.valueOf(r_spd));
```

```
    }           // End of failed speed entry
   tmp_acl = Integer.parseInt(acl_string);
   if ((tmp_acl > MAX_ACL) || (tmp_acl < MIN_ACL))
    {
     acl_good = false;
     acceleration.setBackground(Color.red);
     acceleration.setText(String.valueOf(r_acl));
    }           // End of failed acceration entry
  }             // End of try
 catch (Exception e)
  {

  }             // End of catch
 if ((crs_good == true) && (spd_good == true) && (acl_good == true))
  {
   course.setBackground(Color.white);
   speed.setBackground(Color.white);
   acceleration.setBackground(Color.white);
   course.setText(String.valueOf(tmp_crs));
   r_crs = tmp_crs;
   speed.setText(String.valueOf(tmp_spd));
   r_spd = tmp_spd;
   acceleration.setText(String.valueOf(tmp_acl));
   r_acl = tmp_acl;
  }             // End of good values submitted
 }              // End of read_values

}               // End of class NavUpdate

/** Class compass displays the current RT_RCUSC heading.
*/
class Compass extends Canvas
{
        // VARIABLES for class Compass
        // COMPASS_X_CEN            - Constant integer compass x center
        // COMPASS_Y_CEN            - Constant integer compass y center
        // MAX_Y                    - Constant integer used for drawing needle
        // MAX_X                    - Constant integer used for drawing needle
        // MAX_LIT_Y                - Constant integer used for drawing needle
        // degree                   - Double variable, current compass value
        // degs_rads                - Double variable, radian value of the heading
        // x_values                 - Integer array used to draw needle
        // y_values                 - Integer array used to draw needle
        // pts                      - Integer variable, number of points to draw needle
        // rad_to_deg               - Double variable, conversion factor of degrees to rads
        // outerparent              - Variable indicating RT_RCUSCHpage as parent
        // cos_ret                  - Double variable, cosine of heading
        // sin_ret          - Double variable, sine of heading

 final int COMPASS_X_CEN = 130;
 final int COMPASS_Y_CEN = 110;
 final int MAX_Y = 50;
 final int MAX_LIT_Y = 5;
 final int MAX_X = 10;
```

```
 static double degree = 30.0;
 double degs_rads;
 int x_values[] = {140,130,120,130,140};
 int y_values[] = {110,60,110,113,110};
 int pts = x_values.length;
 double rad_to_deg = Math.PI / 180.0;
 RT_RCUSCHpage outerparent;
 double cos_ret,sin_ret;

 /** Constructor Compass(RT_RCUSCHpge) creates an object of class Compass
  * @param target is of type RT_RCUSCHpage
 */
 Compass(RT_RCUSCHpage target)
  {
   this.outerparent = target;
   setBackground(Color.blue);
  }                  // End of Compass

/** Method set_compass(int) sets new heading for compass.
 * @param compass_value is of type int, sets compass to new value.
*/
public boolean set_compass(int compass_value)
 {
  degree = (double) compass_value;
  return true;
 }                              // Set_compass

 /** Method paint(Graphics) displays current heading.
  * @param g is of type Graphics, use to paint to screen.
 */
 public void paint(Graphics g)
  {
   degs_rads = degree * rad_to_deg;
   cos_ret = Math.cos(degs_rads);
   sin_ret = Math.sin(degs_rads);
   x_values[0] = (int) (Math.round(cos_ret * MAX_X) + COMPASS_X_CEN);
   y_values[0] = (int) (Math.round(sin_ret * MAX_X) + COMPASS_Y_CEN);
   x_values[1] = (int) (Math.round(sin_ret * MAX_Y) + COMPASS_X_CEN);
   y_values[1] = (int) (COMPASS_Y_CEN - Math.round(cos_ret * MAX_Y));
   x_values[2] = (int) (COMPASS_X_CEN - Math.round(cos_ret * MAX_X));
   y_values[2] = (int) (COMPASS_Y_CEN - Math.round(sin_ret * MAX_X));
   x_values[3] = (int) (COMPASS_X_CEN - Math.round(sin_ret * MAX_LIT_Y));
   y_values[3] = (int) (COMPASS_Y_CEN + Math.round(cos_ret * MAX_LIT_Y));
   x_values[4] = x_values[0];
   y_values[4] = y_values[0];
   // Next instruction instantiates a new polygon object - the needle
   Polygon comp_needle = new Polygon(x_values,y_values,pts);
   g.setColor(Color.white);
   g.fillOval(50,30,160,160);
   g.drawString("360",120,25);
   g.drawString("180",120,205);
   g.drawString("270",31,116);
   g.drawString("90",212,116);
   g.setColor(Color.black);
```

```
    g.drawLine(130,30,130,190);
    g.drawLine(50,110,210,110);
    g.fillPolygon(comp_needle);
   }                    // End of paint
}                       // End of class Compass

/** Class Plot plots past current (at grid center) and up to the last 15
 *   RT_RCUSC positions.
*/
class Plot extends Canvas
{
        // VARIABLES for class Plot
        // outerparent          - Variable that indicates that RT_RCUSCHpage is parent
        // MAX_ARRAY            - Constant int - maximum number of points that can be plotted
        // ship_history         - Array of type points - positions of RT_RCUSC
        // last_point           - Number of positions in ship_history
        // x_center             - String variable used to display x value of grid center
        // y_center             - String variable used to display y value of grid center

 RT_RCUSCHpage outerparent;
 static final int MAX_ARRAY = 25;
 static Point[] ship_history = new Point[MAX_ARRAY];
 static int last_point = 0;
 String x_center,y_center;

 /** Constructor Plot(RT_RCUSCHpage) used to create an instance of class Plot.
  * @param target is of type RT_RCUSCHpage
 */
 Plot(RT_RCUSCHpage target)
  {
   this.outerparent = target;
   setBackground(Color.green);
  }                     // End of Compass

 /** Method plot_array(Point[], int) updates RT_RCUSC positions array.
  * @param ship_hist[] of type Point array - last positions of RT_RCUSC.
  * @param last_entry of type int - is the number of postions in array.
 */

 public boolean plot_array(Point ship_hist[], int last_entry)
  {
        // VARIABLES for method plot_array
        // index            - Integer variable used as array index
        // index2           - Integer  variable used as array index
         // keep_looping    - Boolean variable used for loop control

  int index;
  int index2;
  boolean keep_looping = true;
  index2 = (last_entry - 1);
  index = 0;
  while (keep_looping)
   {
    ship_history[index] = ship_hist[index2];
```

```
  if ((index2 <= 0) || (index >= (MAX_ARRAY - 1)))
    keep_looping = false;
  else
   {
    index++;
    index2--;
   }                        // End of else
 }                          // End of while keep_looping
 last_point = index;
 return true;
}               //End of plot_array

/** Method paint(Graphics) plots RT_RCUSC positions to panel.
*/
public void paint(Graphics g)
{
       // VARIABLES for method paint(Graphics)
       // REC_WIDTH          - Constant integer, grid width
       // REC_LENGTH         - Constant integer, grid length
       // index              - Integer variable used for array index
       // prev_x             - Integer variable used for line computations
       // prev_y             - Integer variable used for line computations
       // cur_x              - Integer variable used for line computations
        // cur_y             - integer variable used for line computations

 final int REC_WIDTH = 180;
 final int REC_LENGTH = 180;
 int index,prev_x,prev_y,cur_x,cur_y;
 prev_x = 0;
 prev_y = 0;
 g.setColor(Color.white);
 g.fillRect(50,15,REC_WIDTH,REC_LENGTH);
 g.setColor(Color.black);
 g.drawLine(140,15,140,195);
 g.drawLine(50,105,230,105);
 x_center = Integer.toString(ship_history[0].x);
 y_center = Integer.toString(ship_history[0].y);
 g.drawString(x_center,130,12);
 g.drawString(y_center,20,105);
 for (index = 0; index <= last_point; index++)
  {
  if (index > 0)
   {
   if ((Math.abs(ship_history[index].x) < REC_WIDTH) &&
      (Math.abs(ship_history[index].y) < REC_LENGTH))
    {
    cur_x = ship_history[index].x - ship_history[0].x;
    cur_x = -1 * cur_x;
    cur_y = ship_history[index].y - ship_history[0].y;
    g.setColor(Color.green);
    g.drawLine((140 + prev_y),(105 + prev_x),(140 + cur_y),(105 + cur_x));
    g.setColor(Color.black);
    prev_x = cur_x;
    prev_y = cur_y;
```

```
      }                    // End of boundary check
     }                     // End of if > 0
    }                      // End of for loop
   }                       // End of paint
 }                         // End of class Plot

// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants
 {
 final static int MSG_LENGTH = 36;
 final static int NUM_OF_SHIP_PARMS = 4;   // x pos,y pos,course and speed
 final static int LAT_PARM_POS = 4;
 final static int LON_PARM_POS = 8;
 final static int CRS_PARM_POS = 12;
 final static int SPD_PARM_POS = 16;
 final static int X_POS = 4;
 final static int Y_POS = 8;
 final static int CRS_POS = 12;
 final static int SPD_POS = 16;
 final static int C_CRS_POS = 4;
 final static int C_SPD_POS = 8;
 final static int C_ACL_POS = 12;
 final static int PARM_LENGTH = 4;
 final static byte ZERO_BYTE = 0;


 static byte[] nav_data = new byte[MSG_LENGTH];
 char char_digit;

  /** Nav_Message Constructor for class
 */
 Nav_Message()
  {

  }                  // End of constructor with no arguments


/** Nav_Message(int,int,int,int) Constructor for class - 4 parameters
  * @param srt_lat is starting latitude of RT_RCUSC.
  * @param srt_lon is starting longitude of RT_RCUSC.
  * @param srt_crs is starting course of RT_RCUSC.
  * @param srt_spd is starting speed of RT_RCUSC.
 */
 Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
  {
  int index,index2;
  int num_length,num_128,rem_128;
  for (index = 0; index < MSG_LENGTH; index++)
   nav_data[index] = ZERO_BYTE;          // Zero out message array

  // Code to fill in speed field of nav message
  nav_data[SPD_PARM_POS + 3] = (byte)(0x0000007f & srt_spd);

  // Code to fill in course field of nav message
```

```
   num_128 = srt_crs / 128;
   nav_data[CRS_PARM_POS + 2] = (byte)num_128;
   rem_128 = srt_crs % 128;
   nav_data[CRS_PARM_POS + 3] = (byte)rem_128;

   nav_data[MSG_TYPE_POS] = NAV_TYPE;
   nav_data[REC_TYPE_POS] = NAV_INIT;
   nav_data[SUB_REC_POS] = ZERO_BYTE;
   nav_data[SER_RET_POS] = ZERO_BYTE;

  }                          // End of method Nav_Message

 // Method update_msg updates nav parameters for class
 void update_msg(int[] cmd_update)
  {
  int num_of_128,remainder;
  int num_length;
  int index;
  for (index = 0; index < MSG_LENGTH; index++)
   nav_data[index] = ZERO_BYTE;          // Zero out message array
  nav_data[C_ACL_POS + 3] = (byte)(0x0000007f & cmd_update[2]); //Acceration
  nav_data[C_SPD_POS + 3] = (byte)(0x0000007f & cmd_update[0]); //Speed
  num_of_128 = cmd_update[1] / 128;
  nav_data[C_CRS_POS + 2] = (byte)num_of_128;
  remainder = cmd_update[1] % 128;
  nav_data[C_CRS_POS + 3] = (byte)remainder;
  nav_data[MSG_TYPE_POS] = NAV_TYPE;
  nav_data[REC_TYPE_POS] = NAV_CMD;
  nav_data[SUB_REC_POS] = ZERO_BYTE;
  nav_data[SER_RET_POS] = ZERO_BYTE;

 }                          // End of update_msg

// Method get_nav_msg retrieves nav message
void get_nav_msg(byte[] init_msg)
 {
  int index;
  for (index = 0; index < MSG_LENGTH; index++)
    init_msg[index] = nav_data[index];
 }                          // End of get init_msg

// Method get_ship_info retrieves ship's position and course
void get_ship_info(byte[] rec_buf,int[] ship_info)
 {
 int index,num_128,rem_128;

 // Code to fill in spd field of nav message
 ship_info[3] = 0;
 ship_info[3] = rec_buf[SPD_POS + 3];

 // Code to fill in crs field of nav message
 ship_info[2] = 0;
 num_128 = rec_buf[CRS_POS + 2];
 num_128 = num_128 * 128;
```

```
 rem_128 = rec_buf[CRS_POS + 3];
 ship_info[2] = num_128 + rem_128;

 // Code to fill in y pos field of nav message
 ship_info[1] = 0;
 num_128 = rec_buf[Y_POS + 2];
 num_128 = num_128 * 128;
 rem_128 = rec_buf[Y_POS + 3];
 ship_info[1] = num_128 + rem_128;

 // Code to fill in x pos field of nav message
 ship_info[0] = 0;
 num_128 = rec_buf[X_POS + 2];
 num_128 = num_128 * 128;
 rem_128 = rec_buf[X_POS + 3];
 ship_info[0] = num_128 + rem_128;

 }                          // End of get_ship_info

// Method get_ship_init retrieves ship's initialization info
void get_ship_init(byte[] rec_buf,int[] ship_init_info)
 {
 int index,num_128,rem_128;
 ship_init_info[0] = 0;     // Zero out lat field
 ship_init_info[1] = 0;     // Zero out lon field
 ship_init_info[2] = 0;     // Zero out the course field
 ship_init_info[3] = 0;     // Zero out the speed field

 // Parse out the speed field of nav message
 ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];

 // Parse out the course field of nav message
 num_128 = rec_buf[CRS_PARM_POS + 2];
 num_128 = num_128 * 128;
 rem_128 = rec_buf[CRS_PARM_POS + 3];
 ship_init_info[2] = num_128 + rem_128;

 }                          // End of get_ship_init

// Method get_ship_cmd retrieves ship's command course,speed and acceleration
void get_ship_cmd(byte[] rec_buf,int[] ship_cmd_info)
 {
 int index,num_128,rem_128;
 ship_cmd_info[0] = 0;
 ship_cmd_info[1] = 0;
 ship_cmd_info[2] = 0;

 // Parse out the acceleration  field of nav message
 ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

 // Parse out the speed field of nav message
 ship_cmd_info[1] = rec_buf[C_SPD_POS + 3];

 // Parse out the course field of nav message
```

```
    num_128 = rec_buf[C_CRS_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[C_CRS_POS + 3];
    ship_cmd_info[0] = num_128 + rem_128;

   }                        // End of get_ship_cmd


   // Method build_status_msg parses status message received from server
   void build_status_msg(byte[] nav_data, int[] ship_status_info)
    {
     int index,num_128,rem_128;
     int temp_value;                    // Used to prevent zeroing of ship_status_info
     for (index = 0; index < MSG_LENGTH; index++)
      nav_data[index] = ZERO_BYTE;      // Zero out message array

     // Code to fill in spd position field of nav message
     nav_data[SPD_POS + 3] = (byte)(0x0000007f & ship_status_info[3]);

     // Code to fill in crs position field of nav message
     num_128 = ship_status_info[2] / 128;
     rem_128 = ship_status_info[2] % 128;
     nav_data[CRS_POS + 3] = (byte)rem_128;
     nav_data[CRS_POS + 2] = (byte)num_128;

     // Code to fill in y position field of nav message
     num_128 = ship_status_info[1] / 128;
     rem_128 = ship_status_info[1] % 128;
     nav_data[Y_POS + 3] = (byte)rem_128;
     nav_data[Y_POS + 2] = (byte)num_128;


     // Code to fill in x position field of nav message
     num_128 = ship_status_info[0] / 128;
     rem_128 = ship_status_info[0] % 128;
     nav_data[X_POS + 3] = (byte)rem_128;
     nav_data[X_POS + 2] = (byte)num_128;
     nav_data[MSG_TYPE_POS] = NAV_TYPE;
     nav_data[REC_TYPE_POS] = NAV_CMD;
     nav_data[SUB_REC_POS] = ZERO_BYTE;
     nav_data[SER_RET_POS] = ZERO_BYTE;

    }                       // End of build_status_msg

}                           // End of class Nav_Message

// Class Net_Connection makes socket conection with server
// and exchanges messages with server.
class Net_Connection implements Runnable, Smart_Constants  // Thread
 { //VARIABLES for class Nav_Message
 // i_addr                 - Internet address of client
 // client_sock            - Socket used by client for exchanging data with server
 // client_receiver        - Tread used to decouple receiving and sending data
```

```
// receiver_buffer          - Byte buffer used for receiving data from server
// new_data                 - Variable used to indicate if new data has arrived
 static InetAddress i_addr;
 DatagramSocket client_sock;
 Thread client_receiver;
 static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
 static boolean new_data = false;

 public Net_Connection(InetAddress i_addr)
  {
   this.i_addr = i_addr;
  }                  // End of Net_Connection

 public boolean send_msg(byte[] msg_to_server)
  {
  if (client_sock == null)
   {
   try
    {
    client_sock = new DatagramSocket();
    }
   catch (Exception e)
    {
    System.out.println("Could not create DatagramSocket");
    System.out.println(e);
   }                         // End of try - catch logic
  }                          // End of if null
  DatagramPacket c_packet = new DatagramPacket(msg_to_server,msg_to_server.length,i_addr,
                              SERVER_PORT);
 try
  {
   client_sock.send(c_packet);
  }
 catch (Exception e)
  {
   System.out.println("Error sending packet");
   System.out.println(e);
   return false;
  }                          // End of try - catch block
 return true;
}                            // End of send_msg

public void run()
{
 DatagramPacket rec_packet = new DatagramPacket(receiver_buffer,receiver_buffer.length);
 try
  {
  while (true)
   {
    client_sock.receive(rec_packet);        // Client receives data here
    new_data = true;                        // Might have to implement buffer
   }                                        // End of while
  }                                         // End of try
 catch (Exception e)
```

```
    {
     System.out.println("Could not receive packet");
     System.out.println(e);
     client_receiver = null;
     return;
    }                          // End of try - catch block
  }                            // End of run

  public void stop()
  {
   if (client_receiver != null)
    {
     client_receiver.stop();
    }
   client_sock.close();
  }                                    // End of stop

  public boolean data_receive()
  {
   return new_data;
  }                                    // End of data_receive

  public void clear_data_receive()
  {
   new_data = false;
  }                                    // End of clear_data_receive

  public void get_message(byte[] received_data)
  {
    int index;
    for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
     received_data[index] = receiver_buffer[index];
  }                                    // End of get_message

}                                      // End of class Net_Connection
```

APPENDIX M. RT_SERVER.JAVA FOR UDP

```
// The sever portion of the RT_RCUSC software is made up of four Java classes.
// RT_Sever is the driver of the server position of RT_RCUSC. This class is
// responsible for instantiation of the Model, Net_S_Connection and Nav_Message
// classes. RT_Server also makes periodical calls to the Model class for updates of the
// model's attributes.
import java.awt.*;
import java.net.*;
import java.io.*;
import java.lang.*;

public class RT_Server
 {
 public static void main (String args[])
  {
   new RT_RCUSC_Server();
  }                                          // End of main
 }                                           // End of RT_Server

class RT_RCUSC_Server implements Smart_Constants
 {
 Net_S_Connection server_con;
 Nav_Message nav_msg;
 Model rt_model;
 static boolean keep_looping = true;
 static boolean start_model = false;
 final static int INIT_FIELDS = 4;           // Number of fields in init message
 final static int STATUS_FIELDS = 4;         // Number of fields in statur fields
 final static int CMD_FIELDS = 3;// Number of fields in command message
 final static int TENTH_SEC = 100;           // Tenth of sec counter
 final static int TWO_SEC = 21;              // Twenty tenth = 2 secs
 int two_sec_counter = 0;
 static byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH];    // Byte buffer for receive message
 static byte[] send_buffer = new byte[MAX_MESSAGE_LENGTH];   // Byte buffer for sending message
 static int[] rt_rcusc_init = new int[INIT_FIELDS];          // Integer buffer for initial info
 static int[] rt_rcusc_cmd = new int[CMD_FIELDS];            // Integer buffer for command info
 static int[] rt_rcusc_status = new int[STATUS_FIELDS];      // Integer buffer for status info
 byte rec_type,sub_rec_type;
 FileOutputStream fsnt3_out;                 // Output stream for command received from client
 DataOutputStream dsnt3_out;                 // Provides methods for writing primitive types
 FileOutputStream fsnt_out;                  // Output stream for ship's position and speed
 DataOutputStream dsnt_out;                  // Provides methods for writing primitive types
 String ser_num,crs_string,spd_string,time;



 RT_RCUSC_Server()
  {
  try
   {
    long current_time;
    FileOutputStream fsnt3_out = new FileOutputStream("rec_cmd.txt");
```

```
DataOutputStream dsnt3_out = new DataOutputStream(fsnt3_out);
FileOutputStream fsnt_out = new FileOutputStream("sent_u.txt");
DataOutputStream dsnt_out = new DataOutputStream(fsnt_out);
server_con = new Net_S_Connection(); //Instantiate "network logic class"
nav_msg = new Nav_Message();                    // Instantiate "Naviation logic class"


// Loops until a stop message from client is received
while (keep_looping)
 {
 if (server_con.data_receive())              // Checks if client has sent message
  {
  server_con.clear_data_receive();
  server_con.get_message(rec_buffer);
  rec_type = rec_buffer[MSG_TYPE_POS];
  switch (rec_type)
   {
   case NAV_TYPE:
    {
    sub_rec_type = rec_buffer[REC_TYPE_POS];
    switch (sub_rec_type)
     {
     case NAV_INIT:
      if (start_model == false)
       {
       try
        {
         start_model = true;
         nav_msg.get_ship_init(rec_buffer,rt_rcusc_init);
         rt_model = new Model(rt_rcusc_init[0],rt_rcusc_init[1],rt_rcusc_init[2],rt_rcusc_init[3]);
         ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);
         dsnt3_out.writeBytes(ser_num);
         dsnt3_out.writeBytes("    ");
         crs_string = Integer.toString(rt_rcusc_init[2]);
         dsnt3_out.writeBytes(crs_string);
         spd_string = Integer.toString(rt_rcusc_init[3]);
         dsnt3_out.writeBytes("    ");
         dsnt3_out.writeBytes(spd_string);
         dsnt3_out.writeChar('\n');       // Output sent messages to file
        }                                 // End of try
       catch (IOException ioe)
        {
          System.out.println(" Could not write init to rec_cmd.txt output file");
        }
       }                                  // End of start_model = false
     break;
    case NAV_CMD:

     nav_msg.get_ship_cmd(rec_buffer,rt_rcusc_cmd);  // Parse command
     try
      {
       ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);
       dsnt3_out.writeBytes(ser_num);
       dsnt3_out.writeBytes("    ");
```

```
          crs_string = Integer.toString(rt_rcusc_cmd[0]);
          dsnt3_out.writeBytes(crs_string);
          spd_string = Integer.toString(rt_rcusc_cmd[1]);
          dsnt3_out.writeBytes("    ");
          dsnt3_out.writeBytes(spd_string);
          dsnt3_out.writeChar('\n');    // Output command message received from client
          }                             // End of try
        catch (IOException ioe)
         {
          System.out.println(" Could not write cmd to rec_cmd.txt output file");
         }                              // End of catch
        rt_model.set_cmd_data(rt_rcusc_cmd);
        break;
        default:                .   // Do nothing with nav stat messages
       }                              // End of switch for sub type
     }                                // End of case NAV_TYPE
     break;
    case MODEL_STOP:
     start_model = false;
     keep_looping = false;
     server_con.stop();             // Stop server_con
     try
      {
       fsnt3_out.close();
       dsnt_out.close();
      }
     catch (IOException ioe)
      {
       System.out.println("could not close file");
      }
     break;
      default:                      // Shouldn't be getting her
   }                                // End of switch for type
 }                                  // End of if data_receive
try
 {
  Thread.sleep(TENTH_SEC); // Tenth of a second delay
 }
catch (InterruptedException e) { }      // Sleep exception
if (((two_sec_counter % TWO_SEC) == 0) && (start_model == true))
 {
   two_sec_counter = 1;
   rt_model.ship_status(rt_rcusc_status);
   nav_msg.build_status_msg(send_buffer,rt_rcusc_status);
   ser_num = Byte.toString(send_buffer[SER_NUM_LOC]);
   current_time = System.currentTimeMillis();
   time = Long.toString(current_time);
   dsnt_out.writeBytes(ser_num);
   dsnt_out.writeBytes("    ");
   dsnt_out.writeBytes(time);
   dsnt_out.writeChar('\n');                    // Output sent messages to file
   server_con.send_msg(send_buffer);            // Send rt_rcusc update to client
 }                                              // End if
else
```

```
        two_sec_counter++;
      }                                          // End of while keep_looping
    }                                            // End of opening files
   catch (IOException ioe)
    {
     System.out.println(" Could not open rec_cmd.txt output file");
    }
  }                                  // End of RT_RCUSC_Server constructure

}                                    // End of RT_RCUSC_Server class


/** Class Model updates SmartShip position based on a 3 second interval.
*/
class Model
{
        // VARIABLES for class Model
        // array_index          - Integer variable used as array index
        // ship_x_pos           - Double variable, x position of SmartShip
        // ship_y_pos           - Double variable, y position of SmartShip
        // ship_crs             - Integer variable, SmartShip course
        // ship_spd             - Integer variable, SmartShip speed
        // ship_lat             - Integer variable, SmartShip latitude
        // ship_lon             - Integer variable, SmartShip longitude
        // ship_cmd_crs         - Integer variable, SmartShip command course
        // ship_cmd_spd         - Integer variable, SmartShip command speed
        // ship_acceration- Integer variable, Smartship acceleration
        // deg_per_2sec         - Constant Integer, maximum number of degs ship can turn
        //                      -  in a position update interval
        // max_delta            - Constant integer, sets heading to cmd heading if
        //                      -  within 5 degs
        // full_compass         - Constant integer, number of degrees in a compass
        // half_compass         - Constant integer, number of degrees in half a compass
        // FEET_MILE            - Constant double, number of foots in mile
        // SECS_HOUR            - Constant double, number of seconds in a hour
        // SECS_UPDATE          - Constant double, number of seconds between update
        // UNITS                - Constant double, number of feet each pixel represents

 static int array_index = 0;
 static double ship_x_pos = 0.0;
 static double ship_y_pos = 0.0;
 static int ship_crs = 0;
 static int ship_spd = 0;
 static int ship_lat = 0;
 static int ship_lon = 0;
 static int ship_cmd_crs = 345;
 static int ship_cmd_spd = 20;
 static int ship_acceration = 1;
 final int deg_per_2sec = 5;
 final int max_delta = 355;
 final int full_compass = 360;
 final int half_compass = 180;
 final double FEET_MILE = 5280.0;
 final double SECS_HOUR = 3600.0;
```

```
final double SECS_UPDATE  = 3.0;
final double UNITS = 50.0;

/** Model(int,int,int,int) Constructor for class Model - 4 parameters initial latitude, longitude,
 * @param srt_lat is of type int - starting latitude of Smart Ship.
 * @param srt_lon is of type int - starting longitude of Smart Ship.
 * @param srt_crs is of type int - starting course of Smart Ship at begining of interval.
 * @param srt_spd is of type int - starting speed of Smart Ship at begining of interval.
 * @param ship_cmd_spd of type int - command speed of Smart Ship.
 * @param ship_cmd_crs of type int - command course of Smart Ship.
 *   course and speed.
*/

Model(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
 {
  ship_lat = srt_lat;
  ship_lon = srt_lon;
  ship_crs = srt_crs;
  ship_spd = srt_spd;
  ship_cmd_spd = srt_spd;
  ship_cmd_crs = srt_crs;
 }              // End of Model




/** Method ship_status(int[]) updates current SmartShip position, speed and course.
 * @param status array of int current position ,speed and course of Smart Ship.
*/
public boolean ship_status(int status[])
 {
       // VARIABLES for method ship_status
       // delta_crs              - Integer variable, delta between heading and
       //                        -  command heading
       // rad_to_deg             - Double variable, conversion factor for degrees to radians
       // crs_rads               - Double variable, heading in radians
       // temp_x                 - Double variable, used for computation
       // temp_y                 - Double variable, used for computation
 int delta_crs = 0;
 double rad_to_deg = Math.PI / 180.0;
 double crs_rads;
 double temp_x,temp_y;


 if (ship_crs != ship_cmd_crs)
  {
  delta_crs = Math.abs(ship_crs - ship_cmd_crs);
  if ((delta_crs <= deg_per_2sec) || (delta_crs >= max_delta))
   ship_crs = ship_cmd_crs;
  else
```

```
      {
       delta_crs = ship_cmd_crs - ship_crs;
       if ((delta_crs > 0) && (delta_crs <= half_compass))
        ship_crs = ship_crs + deg_per_2sec;
       else if ((delta_crs > 0) && (delta_crs > half_compass))   //5/6
        ship_crs = ship_crs - deg_per_2sec;
       else if ((delta_crs < 0) && (Math.abs(delta_crs) > half_compass))
        ship_crs = ship_crs + deg_per_2sec;
       else
        ship_crs = ship_crs - deg_per_2sec;
       if (ship_crs > full_compass)
        ship_crs = ship_crs - full_compass;
       if (ship_crs < 0)
        ship_crs = ship_crs + full_compass;
      }                          // End of else delta crs !<= deg_per_2sec
     }                           // End of cmd_crs != ship_crs
    if (ship_spd != ship_cmd_spd)
     {
      if (ship_spd > ship_cmd_spd)
       ship_spd = ship_spd - ship_acceration;
      else
       ship_spd = ship_spd + ship_acceration;
     }                           // End of ship_spd != ship_cmd_spd
    crs_rads = ((double) ship_crs * rad_to_deg);
    temp_x = (((Math.cos(crs_rads) * (double) ship_spd * FEET_MILE) /
          (SECS_HOUR * SECS_UPDATE)) / UNITS);
    temp_y = (((Math.sin(crs_rads) * (double) ship_spd * FEET_MILE) /
          (SECS_HOUR * SECS_UPDATE)) / UNITS);
    ship_x_pos = ship_x_pos + temp_x;
    ship_y_pos = ship_y_pos + temp_y;
    status[0] = (int) Math.round(ship_x_pos);
    status[1] = (int) Math.round(ship_y_pos);
    status[2] = ship_crs;
    status[3] = ship_spd;
    return true;
    }                           // End of ship_status

   /** Method set_cmd_data(int[]) updates model object with  new
    *   command speed and heading.
    * @param cmd_info array of int containing current command data for Smart Ship.
   */
   public boolean set_cmd_data(int cmd_info[])
    {
     ship_cmd_spd = cmd_info[1];
     ship_cmd_crs = cmd_info[0];
     return true;
    }                  // End of set_cmd_data

  }                    // End of class model


// Class Net_S_Connection blocks for socket connection with client,
// sends and receives data with the client.
class Net_S_Connection implements Runnable, Smart_Constants
```

```
{
// VARIABLES for class Net_S_Connection
// client_addr          - URL address for computer running applet
// server_sock          - Server socket used to establish socket with client
// server_receiver      - Thread for decoupling receiving and sending data
// input_data           - Input stream used to receive data from client
// output_data          - Output stream used to send data to client
// server_receiver      - Thread used to decouple receiving and sending data
// client_port          - Port address blocking for client
// ser_num              - Serial number of a message
// receiver_buffer      - Byte buffer to hold input and output messages
// new_data             - Variable indicating arrival of new data
// fsn2_out             - File used to output STAT messages from client
// dsnt2_out            - Used to output STAT messages from client

static InetAddress client_addr;
DatagramSocket server_sock;
Thread server_receiver;
int client_port;
long current_time;
String time,ser_num;
static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
static boolean new_data = false;
FileOutputStream fsnt2_out;
DataOutputStream dsnt2_out;

// Constructor method for class Net_S_Connection
public Net_S_Connection()
 {
  if (server_receiver == null)
   {
    server_receiver = new Thread(this);
    server_receiver.start();
   }              // End of server_receiver
 }                // End of Net_S_Connection


// Task that blocks until client establishes socket with server
//  then receives messages from client.
 public void run()
 {
  try
   {
    server_sock = new DatagramSocket(SERVER_PORT);
   }              // End of try
  catch (Exception e)
   {
    System.out.println("Could not create DatagramSocket");
    System.out.println(e);
    System.exit(0);
   }              // End of catch
  try
   {
   FileOutputStream fsnt2_out = new FileOutputStream("stat_u.txt");
```

```
      DataOutputStream dsnt2_out = new DataOutputStream(fsnt2_out);
      DatagramPacket rec_packet = new DatagramPacket(receiver_buffer,receiver_buffer.length);
      try
       {
       while (true)
        {
        try
         {
          server_sock.receive(rec_packet);     // Receive data from client
         }                    // End of try
        catch(IOException e)
         {
          System.out.println("DatagramSocket could not receive a packet");
          System.out.println(e);
          System.exit(0);
         }                    // End of try catch logic
        if (receiver_buffer[REC_SUB_TYPE_POS] != NAV_STAT)
         {
          new_data = true;
          client_addr = rec_packet.getAddress();
          client_port = rec_packet.getPort();
         }                    // End of if not NAV_STAT
        else
         {
          ser_num = Byte.toString(receiver_buffer[SER_NUM_LOC]);
          current_time = System.currentTimeMillis();
          time = Long.toString(current_time);
          dsnt2_out.writeBytes(ser_num);
          dsnt2_out.writeBytes("    ");
          dsnt2_out.writeBytes(time);
          dsnt2_out.writeChar('\n');          // Output sent messages to file
         }                    // Else print out info to file
        }                     // End of while
       }                      // End of try
      catch (Exception e)
       {
        System.out.println("Could not receive packet");
        System.out.println(e);
        server_receiver = null;
        return;
       }           // End of try - catch block
     }             // End of open file
    catch (IOException ioe)
     {
      System.out.println(" Could not open sent_u.txt output file");
     }           // End of catch
    }            // End of run

   // Method send_msg sends ship's poition and heading
   // to client.
   public boolean send_msg(byte[] msg_to_server)
    {
     DatagramPacket c_packet = new
DatagramPacket(msg_to_server,msg_to_server.length,client_addr,client_port);
```

```
  try
   {
    server_sock.send(c_packet);
   }                          // End of try
  catch (Exception e)
   {
    System.out.println("Error sending packet");
    System.out.println(e);
    return false;
   }                          // End of try - catch block
  return true;
 }                           // End of send_msg

 //Method to unplug socket
 public void stop()
  {
  if (server_receiver != null)
   {
    server_receiver.stop();
   }                          // End of if not null
  server_sock.close();
  try
   {
    fsnt2_out.close();
   }                          // End of try
  catch (IOException ioe)
   {
    System.out.println("could not close file");
   }                          // End of catch
 }                            // End of stop

 // Method data_receive indicates if new data has been received.
 public boolean data_receive()
  {
  return new_data;
  }                           // End of data_receive

 // Method clear_data clears new data variable
 public void clear_data_receive()
  {
  new_data = false;
  }                           // End of clear_data_receive

 // Method get_message retrives data received from client for
 //  processing.
 public void get_message(byte[] received_data)
  {
  int index;
  for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
   received_data[index] = receiver_buffer[index];
  }                           // End of get_message

}                             // End of class Net_S_Connection
```

```
// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants
 {
 // VARIABLES for class Nav_Message
 // MSG_LENGTH              - Max length for nav message is 36 bytes
 // NUM_OF_SHIP_PARMS       - Nav message contains 4 parameters for ship
 // LAT_PARM_POS            - Latitude parameter position in byte buffer
 // LON_PARM_POS            - Longitude parameter position in byte buffer
 // CRS_PARM_POS            - Course parameter position in byte buffer
 // SPD_PARM_POS            - Speed parameter position in byte buffer
 // Y_POS                   - Ship's y position in byte buffer
 // X_POS                   - Ship's x position in byte buffer
 // CRS_POS                 - Ship's course position in byte buffer
 // SPD_POS                 - Ship's speed position in byte buffer
 // C_CRS_POS               - Ship's command course position in byte buffer
 // C_SPD_POS               - Ship's command speed position in byte buffer
 // C_ACL_POS               - Ship's command acceleration in byte buffer
 // PARM_LENGTH             - Parameter's length - 4 bytes
 // ZERO_BYTE               - Byte of 0's
 // index2                  - Used for loop variable
 // ser_number              - Serial number of message
 // MAX_SERIAL              - Largest serial number after which value is set to 0
 // nav_data                - Byte buffer used for storing messages
 // char_digit              - Character used to store a digit as a character
  final static int MSG_LENGTH = 36;
  final static int NUM_OF_SHIP_PARMS = 4;    //x pos,y pos,course and speed
  final static int LAT_PARM_POS = 4;
  final static int LON_PARM_POS = 8;
  final static int CRS_PARM_POS = 12;
  final static int SPD_PARM_POS = 16;
  final static int X_POS = 4;
  final static int Y_POS = 8;
  final static int CRS_POS = 12;
  final static int SPD_POS = 16;
  final static int C_CRS_POS = 4;
  final static int C_SPD_POS = 8;
  final static int C_ACL_POS = 12;
  final static int PARM_LENGTH = 4;
  final static byte ZERO_BYTE = 0;
  int index2;
  static byte ser_number = 0;
  static final byte MAX_SERIAL= 125;
  static byte[] nav_data = new byte[MSG_LENGTH];
  char char_digit;

 /** Nav_Message Constructor for class
  */
 Nav_Message()
  {

  }                          // Constructor with no arguments
```

```
/** Nav_Message(int,int,int,int) Constructor for class - 4 parameters
 * @param srt_lat is starting latitude of Smart Ship.
 * @param srt_lon is starting longitude of Smart Ship.
 * @param srt_crs is starting course of Smart Ship.
 * @param srt_spd is starting speed of Smart Ship.
*/
Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
 {
  int index,index2;
  int num_length,num_128,rem_128;

  for (index = 0; index < MSG_LENGTH; index++)
   nav_data[index] = ZERO_BYTE;          // Zero out message array

  // Code to fill in speed field of nav message
  nav_data[SPD_PARM_POS + 3] = (byte)(0x0000007f & srt_spd);

  // Code to fill in course field of nav message
  num_128 = srt_crs / 128;
  rem_128 = srt_crs % 128;
  nav_data[CRS_PARM_POS + 3] = (byte)rem_128;
  nav_data[CRS_PARM_POS + 2] = (byte)num_128;
  nav_data[MSG_TYPE_POS] = NAV_TYPE;
  nav_data[REC_TYPE_POS] = NAV_INIT;
  nav_data[SUB_REC_POS] = ZERO_BYTE;
  nav_data[SER_RET_POS] = ZERO_BYTE;
 }                          // End of method Nav_Message

// Method update_msg updates nav parameters for class
void update_msg(int[] cmd_update)
 {
  int index,index2,num_128,rem_128;
  int num_length;
  for (index = 0; index < MSG_LENGTH; index++)
   nav_data[index] = ZERO_BYTE;          //zero out message array

  // Code to fill in acceleration field of nav message
  nav_data[C_ACL_POS + 3] = (byte)(0x0000007f & cmd_update[2]);

  // Code to fill in speed field on nav message
  nav_data[C_SPD_POS + 3] = (byte)(0x0000007f & cmd_update[1]);

  // Code to fill in crs field of nav message
  num_128 = cmd_update[0] / 128;
  rem_128 = cmd_update[0] % 128;
  nav_data[C_CRS_POS + 3] = (byte)rem_128;
  nav_data[C_CRS_POS + 2] = (byte)num_128;
  nav_data[MSG_TYPE_POS] = NAV_TYPE;
  nav_data[REC_TYPE_POS] = NAV_CMD;
  nav_data[SUB_REC_POS] = ZERO_BYTE;
  nav_data[SER_RET_POS] = ZERO_BYTE;
```

```
 }                          // End of update_msg

// Method get_nav_msg retrieves ship's current data
void get_nav_msg(byte[] init_msg)
 {
  int index;
  for (index = 0; index < MSG_LENGTH; index++)
   init_msg[index] = nav_data[index];
 }                          // End of get init_msg

// Method get_ship_init retrieves ship's initial data
void get_ship_init(byte[] rec_buf,int[] ship_init_info)
 {
  int index,num_128,rem_128;
  ship_init_info[0] = 0;   // Zero out lat field
  ship_init_info[1] = 0;   // Zero out lon field
  ship_init_info[2] = 0;   // Zero out the course field
  ship_init_info[3] = 0;   // Zero out the speed field

  // Parse out the speed field of nav message
  ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];

  // Parse out the course field of nav message
  num_128 = rec_buf[CRS_PARM_POS + 2];
  num_128 = num_128 * 128;
  rem_128 = rec_buf[CRS_PARM_POS + 3];
  ship_init_info[2] = num_128 + rem_128;

 }                  // End of get_ship_init

//Method get_ship_cmd  retrieves ship's command data
void get_ship_cmd(byte[] rec_buf,int[] ship_cmd_info)
 {
  int index,num_128,rem_128;
  ship_cmd_info[0] = 0;
  ship_cmd_info[1] = 0;
  ship_cmd_info[2] = 0;

  // Parse out the acceleration  field of nav message
  ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

  // Parse out the speed field of nav message
  ship_cmd_info[1] = rec_buf[C_CRS_POS + 3];

  // Parse out the course field of nav message
  num_128 = rec_buf[C_SPD_POS + 2];
  num_128 = num_128 * 128;
  rem_128 = rec_buf[C_SPD_POS + 3];
  ship_cmd_info[0] = num_128 + rem_128;

 }                  // End of get_ship_cmd


 // Method build_status_msg builds status message to be sent to client
```

```
void build_status_msg(byte[] nav_data, int[] ship_status_info)
 {
 int index;
 byte num_128,rem_128;
 for (index = 0; index < MSG_LENGTH; index++)
  nav_data[index] = ZERO_BYTE;          //zero out message array

 // Code to fill in spd position field of nav message
 nav_data[SPD_POS + 3] = (byte)(0x0000007f & ship_status_info[3]);

 // Code to fill in crs position field of nav message
 num_128 = (byte)(ship_status_info[2] / 128);
 rem_128 = (byte)(ship_status_info[2] % 128);
 nav_data[CRS_POS + 2] = num_128;
 nav_data[CRS_POS + 3] = rem_128;

 // Code to fill in y position field of nav message
 num_128 = (byte)(ship_status_info[1] / 128);
 rem_128 = (byte)(ship_status_info[1] % 128);
 nav_data[Y_POS + 2] = num_128;
 -nav_data[Y_POS + 3] = rem_128;

 // Code to fill in x position field of nav message
 num_128 = (byte)(ship_status_info[0] / 128);
 rem_128 = (byte)(ship_status_info[0] % 128);
 nav_data[X_POS + 2] = num_128;
 nav_data[X_POS + 3] = rem_128;

 nav_data[MSG_TYPE_POS] = NAV_TYPE;
 nav_data[REC_TYPE_POS] = NAV_CMD;
 nav_data[SUB_REC_POS] = ZERO_BYTE;
 // Fill in serial number
 nav_data[SER_NUM_LOC] = ser_number;
 ser_number++;
 if (ser_number > MAX_SERIAL)
  ser_number = 0;

 }                          // End of build_status_msg

}                           // End of class Nav_Message
```

APPENDIX N. SMART_CONSTANTS.JAVA FOR UDP

```
public interface Smart_Constants
 {
  final static int BITS_IN_BYTE = 8;          // Number of bits in a byte
  final static int BYTES_IN_INT = 4;          // Number of bytes in a integer
  final static int SER_RET_POS = 3;           // Serial number position in buffer
  final static int MSG_TYPE_POS = 0;          // Message type position in buffer
  final static int REC_TYPE_POS = 1;          // Record type position in buffer
  final static int SUB_REC_POS = 2;           // Sub record type position in buffer
  final static int MAX_MESSAGE_LENGTH = 72; // Maximum bytes in buffer
  final static int MAX_ARRAY = 25;
  final static int SERVER_PORT = 1998;    // Serial port used for sockets
  final static int REC_SUB_TYPE_POS = 1;          // Sub record type position in buffer
  final static byte NAV_TYPE = 1;             // NAV_TYPE value
  final static byte NAV_INIT = 1;             // NAV_INIT value
  final static byte NAV_CMD  = 2;             // NAV_CMD value
  final static byte NAV_STAT = 3;             // NAV_STAT value
  final static byte SER_NUM_LOC = 3;      // SER_NUM_LOC type position in buffer
  final static byte MODEL_STOP = 99;      // MODEL_STOP value
}                   // End of Smart_Constants
```

APPENDIX O. EXPERIMENT 1 DATA FOR TCP

Experiment 1 (Server Control using TCP, Windows NT and Internet Explorer)

N	stat_u.txt Message Counter	stat_u.txt	sent_u.txt Message Counter	sent_u..txt	Round Trip Latency Time	Calculated one way Latency time	Freq. Bin
1	0	868731829592	0	868731829512	80	40	0
2	1	868731831755	1	868731831655	100	50	6
3	2	868731833798	2	868731833768	30	15	11
4	3	868731835991	3	868731835891	100	50	16
5	4	868731838034	4	868731838004	30	15	21
6	5	868731840197	5	868731840127	70	35	26
7	6	868731842380	6	868731842290	90	45	31
8	7	868731844493	7	868731844423	70	35	36
9	8	868731846636	8	868731846536	100	50	41
10	9	868731848699	9	868731848659	40	20	46
11	10	868731850852	10	868731850772	80	40	51
12	11	868731852905	11	868731852885	20	10	56
13	12	868731855068	12	868731854998	70	35	61
14	13	868731857221	13	868731857121	100	50	
15	14	868731859284	14	868731859234	50	25	
16	15	868731861438	15	868731861357	81	40.5	
17	16	868731863490	16	868731863470	20	10	
18	17	868731865664	17	868731865583	81	40.5	
19	18	868731867807	18	868731867707	100	50	
20	19	868731869860	19	868731869830	30	15	
21	20	868731872013	20	868731871943	70	35	
22	21	868731874066	21	868731874056	10	5	
23	22	868731876219	22	868731876169	50	25	
24	23	868731878352	23	868731878282	70	35	
25	24	868731880475	24	868731880405	70	35	
26	25	868731882598	25	868731882518	80	40	
27	26	868731884731	26	868731884641	90	45	
28	27	868731886854	27	868731886764	90	45	
29	28	868731888967	28	868731888877	90	45	
30	29	868731891100	29	868731891000	100	50	
31	30	868731893213	30	868731893133	80	40	
32	31	868731895336	31	868731895266	70	35	
33	32	868731897399	32	868731897379	20	10	
34	33	868731899552	33	868731899502	50	25	
35	34	868731901715	34	868731901615	100	50	
36	35	868731903778	35	868731903738	40	20	
37	36	868731905931	36	868731905851	80	40	
38	37	868731908105	37	868731907994	111	55.5	
39	38	868731910158	38	868731910107	51	25.5	
40	39	868731912321	39	868731912231	90	45	
41	40	868731914374	40	868731914344	30	15	
42	41	868731916497	41	868731916467	30	15	

43	42	868731918600	42	868731918580	20	10
44	43	868731920743	43	868731920693	50	25
45	44	868731922866	44	868731922806	60	30
46	45	868731925029	45	868731924929	100	50
47	46	868731927172	46	868731927092	80	40
48	47	868731929295	47	868731929225	70	35
49	48	868731931408	48	868731931338	70	35
50	49	868731933561	49	868731933461	100	50
51	50	868731935654	50	868731935574	80	40
52	51	868731937777	51	868731937687	90	45
53	52	868731939910	52	868731939830	80	40
54	53	868731942023	53	868731941943	80	40
55	54	868731944136	54	868731944056	80	40
56	55	868731946269	55	868731946179	90	45
57	56	868731948383	56	868731948302	81	40.5
58	57	868731950506	57	868731950415	91	45.5
59	58	868731952649	58	868731952549	100	50
60	59	868731954702	59	868731954662	40	20
61	60	868731956855	60	868731956775	80	40
62	61	868731958918	61	868731958888	30	15
63	62	868731961051	62	868731961011	40	20
64	63	868731963204	63	868731963124	80	40
65	64	868731965317	64	868731965237	80	40
66	65	868731967450	65	868731967360	90	45
67	66	868731969503	66	868731969473	30	15
68	67	868731971636	67	868731971586	50	25
69	68	868731973759	68	868731973719	40	20
70	69	868731975912	69	868731975832	80	40
71	70	868731977965	70	868731977955	10	5
72	71	868731980118	71	868731980068	50	25
73	72	868731982281	72	868731982181	100	50
74	73	868731984334	73	868731984314	20	10
75	74	868731986507	74	868731986427	80	40
76	75	868731988640	75	868731988540	100	50
77	76	868731990783	76	868731990693	90	45
78	77	868731992836	77	868731992806	30	15
79	78	868731994990	78	868731994919	71	35.5
80	79	868731997153	79	868731997063	90	45
81	80	868731999206	80	868731999176	30	15
82	81	868732001359	81	868732001289	70	35
83	82	868732003472	82	868732003412	60	30
84	83	868732005615	83	868732005525	90	45
85	84	868732007678	84	868732007638	40	20
86	85	868732009831	85	868732009751	80	40
87	86	868732011884	86	868732011874	10	5
88	87	868732014027	87	868732013987	40	20
89	88	868732016150	88	868732016100	50	25
90	89	868732018273	89	868732018213	60	30
91	90	868732020386	90	868732020336	50	25
92	91	868732022509	91	868732022449	60	30
93	92	868732024662	92	868732024562	100	50

94	93	868732026705	93	868732026675	30	15
95	94	868732028818	94	868732028798	20	10
96	95	868732030971	95	868732030911	60	30
97	96	868732033134	96	868732033034	100	50
98	97	868732035187	97	868732035157	30	15
99	98	868732037340	98	868732037270	70	35
100	99	868732039494	99	868732039423	71	35.5
101	100	868732041546	100	868732041536	10	5
102	101	868732043720	101	868732043650	70	35
103	102	868732045873	102	868732045773	100	50
104	103	868732047926	103	868732047886	40	20
105	104	868732050099	104	868732050009	90	45
106	105	868732052242	105	868732052142	100	50
107	106	868732054305	106	868732054265	40	20
108	107	868732056458	107	868732056378	80	40
109	108	868732058521	108	868732058491	30	15
110	109	868732060684	109	868732060614	70	35
111	110	868732062817	110	868732062727	90	45
112	111	868732064940	111	868732064850	90	45
113	112	868732067053	112	868732066963	90	45
114	113	868732069186	113	868732069086	100	50
115	114	868732071299	114	868732071209	90	45
116	115	868732073412	115	868732073322	90	45
117	116	868732075545	116	868732075435	110	55
118	117	868732077588	117	868732077558	30	15
119	118	868732079741	118	868732079681	60	30
120	119	868732081804	119	868732081794	10	5
121	120	868732083967	120	868732083907	60	30
122	121	868732086121	121	868732086030	91	45.5
123	122	868732088184	122	868732088143	41	20.5
124	123	868732090337	123	868732090257	80	40
125	124	868732092390	124	868732092380	10	5
126	125	868732094553	125	868732094493	60	30
127	0	868732096706	0	868732096616	90	45
128	1	868732098759	1	868732098729	30	15
129	2	868732100912	2	868732100852	60	30
130	3	868732103075	3	868732102985	90	45
131	4	868732105128	4	868732105098	30	15
132	5	868732107281	5	868732107221	60	30
133	6	868732109444	6	868732109344	100	50
134	7	868732111477	7	868732111457	20	10
135	8	868732113600	8	868732113570	30	15
136	9	868732115733	9	868732115693	40	20
137	10	868732117896	10	868732117816	80	40
138	11	868732120059	11	868732119969	90	45
139	12	868732122112	12	868732122082	30	15
140	13	868732124245	13	868732124195	50	25
141	14	868732126368	14	868732126318	50	25
142	15	868732128512	15	868732128431	81	40.5
143	16	868732130635	16	868732130544	91	45.5
144	17	868732132768	17	868732132668	100	50

145	18	868732134821	18	868732134791	30	15
146	19	868732136974	19	868732136904	70	35
147	20	868732139127	20	868732139037	90	45
148	21	868732141190	21	868732141160	30	15
149	22	868732143373	22	868732143273	100	50
150	23	868732145506	23	868732145416	90	45
151	24	868732147559	24	868732147539	20	10
152	25	868732149702	25	868732149652	50	25
153	26	868732151865	26	868732151775	90	45
154	27	868732153908	27	868732153888	20	10
155	28	868732156051	28	868732156011	40	20
156	29	868732158134	29	868732158124	10	5
157	30	868732160297	30	868732160237	60	30
158	31	868732162460	31	868732162360	100	50
159	32	868732164503	32	868732164473	30	15
160	33	868732166636	33	868732166596	40	20
161	34	868732168759	34	868732168709	50	25
162	35	868732170902	35	868732170822	80	40
163	36	868732173036	36	868732172945	91	45.5
164	37	868732175159	37	868732175068	91	45.5
165	38	868732177212	38	868732177182	30	15
166	39	868732179365	39	868732179305	60	30
167	40	868732181518	40	868732181418	100	50
168	41	868732183571	41	868732183541	30	15
169	42	868732185734	42	868732185654	80	40
170	43	868732187787	43	868732187777	10	5
171	44	868732189950	44	868732189890	60	30
172	45	868732192073	45	868732192003	70	35
173	46	868732194226	46	868732194126	100	50
174	47	868732196279	47	868732196259	20	10
175	48	868732198442	48	868732198372	70	35
176	49	868732200585	49	868732200495	90	45
177	50	868732202658	50	868732202608	50	25
178	51	868732204811	51	868732204731	80	40
179	52	868732206874	52	868732206844	30	15
180	53	868732209007	53	868732208957	50	25
181	54	868732211110	54	868732211080	30	15
182	55	868732213233	55	868732213193	40	20
183	56	868732215356	56	868732215306	50	25
184	57	868732217479	57	868732217419	60	30
185	58	868732219613	58	868732219532	81	40.5
186	59	868732221736	59	868732221645	91	45.5
187	60	868732223839	60	868732223769	70	35
188	61	868732225962	61	868732225882	80	40
189	62	868732228015	62	868732227995	20	10
190	63	868732230178	63	868732230108	70	35
191	64	868732232341	64	868732232241	100	50
192	65	868732234454	65	868732234354	100	50
193	66	868732236577	66	868732236477	100	50
194	67	868732238700	67	868732238610	90	45
195	68	868732240823	68	868732240733	90	45

196	69	868732242946	69	868732242846	100	50
197	70	868732245069	70	868732244969	100	50
198	71	868732247122	71	868732247082	40	20
199	72	868732249285	72	868732249215	70	35
200	73	868732251348	73	868732251338	10	5
201	74	868732253501	74	868732253451	50	25
202	75	868732255654	75	868732255564	90	45
203	76	868732257707	76	868732257677	30	15
204	77	868732259850	77	868732259790	60	30
205	78	868732262014	78	868732261913	101	50.5
206	79	868732264046	79	868732264026	20	10
207	80	868732266149	80	868732266139	10	5
208	81	868732268283	81	868732268262	21	10.5
209	82	868732270416	82	868732270376	40	20
210	83	868732272529	83	868732272489	40	20
211	84	868732274662	84	868732274602	60	30
212	85	868732276835	85	868732276735	100	50
213	86	868732278888	86	868732278848	40	20
214	87	868732281051	87	868732280971	80	40
215	88	868732283214	88	868732283114	100	50
216	89	868732285267	89	868732285227	40	20
217	90	868732287420	90	868732287340	80	40
218	91	868732289563	91	868732289463	100	50
219	92	868732291616	92	868732291576	40	20
220	93	868732293769	93	868732293689	80	40
221	94	868732295882	94	868732295812	70	35
222	95	868732298005	95	868732297925	80	40
223	96	868732300148	96	868732300058	90	45
224	97	868732302201	97	868732302171	30	15
225	98	868732304364	98	868732304284	80	40
226	99	868732306507	99	868732306417	90	45
227	100	868732308631	100	868732308530	101	50.5
228	101	868732310734	101	868732310643	91	45.5
229	102	868732312867	102	868732312756	111	55.5
230	103	868732314900	103	868732314890	10	5
231	104	868732317013	104	868732317003	10	5
232	105	868732319146	105	868732319116	30	15
233	106	868732321269	106	868732321229	40	20
234	107	868732323392	107	868732323352	40	20
235	108	868732325535	108	868732325465	70	35
236	109	868732327698	109	868732327598	100	50
237	110	868732329751	110	868732329711	40	20
238	111	868732331884	111	868732331834	50	25
239	112	868732333997	112	868732333947	50	25
240	113	868732336150	113	868732336060	90	45
241	114	868732338193	114	868732338183	10	5
242	115	868732340346	115	868732340296	50	25
243	116	868732342499	116	868732342409	90	45
244	117	868732344642	117	868732344542	100	50
245	118	868732346685	118	868732346655	30	15
246	119	868732348858	119	868732348768	90	45

247	120	868732351001	120	868732350901	100	50
248	121	868732353074	121	868732353014	60	30
249	122	868732355208	122	868732355127	81	40.5
250	123	868732357381	123	868732357311	70	35
251	124	868732359444	124	868732359424	20	10
252	125	868732361587	125	868732361547	40	20
253	0	868732363700	0	868732363660	40	20
254	1	868732365853	1	868732365773	80	40
255	2	868732368016	2	868732367926	90	45
256	3	868732370079	3	868732370039	40	20
257	4	868732372242	4	868732372152	90	45
258	5	868732374375	5	868732374285	90	45
259	6	868732376488	6	868732376398	90	45
260	7	868732378531	7	868732378521	10	5
261	8	868732380704	8	868732380654	50	25
262	9	868732382857	9	868732382767	90	45
263	10	868732384910	10	868732384890	20	10
264	11	868732387043	11	868732387013	30	15
265	12	868732389166	12	868732389126	40	20
266	13	868732391299	13	868732391239	60	30
267	14	868732393422	14	868732393372	50	25
268	15	868732395546	15	868732395485	61	30.5
269	16	868732397699	16	868732397598	101	50.5
270	17	868732399762	17	868732399712	50	25
271	18	868732401925	18	868732401835	90	45
272	19	868732404078	19	868732403988	90	45
273	20	868732406191	20	868732406101	90	45
274	21	868732408404	21	868732408304	100	50
275	22	868732410457	22	868732410417	40	20
276	23	868732412580	23	868732412540	40	20
277	24	868732414723	24	868732414653	70	35
278	25	868732416876	25	868732416786	90	45
279	26	868732418949	26	868732418899	50	25
280	27	868732421082	27	868732421012	70	35
281	28	868732423235	28	868732423135	100	50
282	29	868732425288	29	868732425248	40	20
283	30	868732427441	30	868732427371	70	35
284	31	868732429514	31	868732429484	30	15
285	32	868732431677	32	868732431597	80	40
286	33	868732433831	33	868732433730	101	50.5
287	34	868732435853	34	868732435843	10	5
288	35	868732437977	35	868732437956	21	10.5
289	36	868732440120	36	868732440070	50	25
290	37	868732442283	37	868732442183	100	50
291	38	868732444436	38	868732444346	90	45
292	39	868732446489	39	868732446459	30	15
293	40	868732449163	40	868732449133	30	15
294	41	868732451316	41	868732451266	50	25
295	42	868732453479	42	868732453379	100	50
296	43	868732455532	43	868732455502	30	15
297	44	868732457685	44	868732457625	60	30

298	45	868732459818	45	868732459738	80	40
299	46	868732461941	46	868732461851	90	45
300	47	868732464064	47	868732463964	100	50
301	48	868732466117	48	868732466087	30	15
302	49	868732468260	49	868732468200	60	30
303	50	868732470423	50	868732470323	100	50
304	51	868732472486	51	868732472446	40	20
305	52	868732474659	52	868732474559	100	50
306	53	868732476712	53	868732476682	30	15
307	54	868732478885	54	868732478795	90	45
308	55	868732480928	55	868732480908	20	10
309	56	868732483091	56	868732483021	70	35
310	57	868732485224	57	868732485144	80	40
311	58	868732487338	58	868732487257	81	40.5
312	59	868732489491	59	868732489400	91	45.5
313	60	868732491544	60	868732491514	30	15
314	61	868732493707	61	868732493627	80	40
315	62	868732495860	62	868732495770	90	45
316	63	868732497983	63	868732497893	90	45
317	64	868732500116	64	868732500026	90	45
318	65	868732502159	65	868732502139	20	10
319	66	868732504282	66	868732504252	30	15
320	67	868732506405	67	868732506365	40	20
321	68	868732508558	68	868732508478	80	40
322	69	868732510681	69	868732510591	90	45
323	70	868732512814	70	868732512724	90	45
324	71	868732514867	71	868732514837	30	15
325	72	868732516990	72	868732516950	40	20
326	73	868732519123	73	868732519073	50	25
327	74	868732521196	74	868732521186	10	5
328	75	868732523359	75	868732523299	60	30
329	76	868732525512	76	868732525422	90	45
330	77	868732527545	77	868732527535	10	5
331	78	868732529668	78	868732529648	20	10
332	79	868732531811	79	868732531761	50	25
333	80	868732533975	80	868732533874	101	50.5
334	81	868732536028	81	868732535997	31	15.5
335	82	868732538181	82	868732538111	70	35
336	83	868732540304	83	868732540224	80	40
337	84	868732542427	84	868732542337	90	45
338	85	868732544480	85	868732544460	20	10
339	86	868732546633	86	868732546573	60	30
340	87	868732548796	87	868732548696	100	50
341	88	868732550839	88	868732550809	30	15
342	89	868732552962	89	868732552922	40	20
343	90	868732555075	90	868732555035	40	20
344	91	868732557218	91	868732557158	60	30
345	92	868732559341	92	868732559271	70	35
346	93	868732561464	93	868732561384	80	40
347	94	868732563607	94	868732563507	100	50
348	95	868732565720	95	868732565630	90	45

349	96	868732567853	96	868732567753	100	50
350	97	868732569896	97	868732569866	30	15
351	98	868732572009	98	868732571979	30	15
352	99	868732574142	99	868732574092	50	25
353	100	868732576275	100	868732576205	70	35
354	101	868732578368	101	868732578328	40	20
355	102	868732580501	102	868732580441	60	30
356	103	868732582645	103	868732582554	91	45.5
357	104	868732584697	104	868732584667	30	15
358	105	868732586851	105	868732586791	60	30
359	106	868732588984	106	868732588904	80	40
360	107	868732591037	107	868732591017	20	10
361	108	868732593200	108	868732593140	60	30
362	109	868732595353	109	868732595263	90	45
363	110	868732597406	110	868732597376	30	15
364	111	868732599559	111	868732599489	70	35
365	112	868732601712	112	868732601622	90	45
366	113	868732603775	113	868732603735	40	20
367	114	868732605938	114	868732605848	90	45
368	115	868732607981	115	868732607971	10	5
369	116	868732610144	116	868732610084	60	30
370	117	868732612307	117	868732612207	100	50
371	118	868732614360	118	868732614330	30	15
372	119	868732616513	119	868732616443	70	35
373	120	868732618656	120	868732618556	100	50
374	121	868732620749	121	868732620669	80	40
375	122	868732622892	122	868732622792	100	50
376	123	868732624955	123	868732624905	50	25
377	124	868732627108	124	868732627028	80	40
378	125	868732629272	125	868732629191	81	40.5
379	0	868732631395	0	868732631305	90	45
380	1	868732633518	1	868732633428	90	45
381	2	868732635661	2	868732635561	100	50
382	3	868732637694	3	868732637674	20	10
383	4	868732639807	4	868732639787	20	10
384	5	868732641940	5	868732641900	40	20
385	6	868732644063	6	868732644023	40	20
386	7	868732646176	7	868732646136	40	20
387	8	868732648299	8	868732648249	50	25
388	9	868732650412	9	868732650362	50	25
389	10	868732652535	10	868732652485	50	25
390	11	868732654678	11	868732654608	70	35
391	12	868732656811	12	868732656721	90	45
392	13	868732658864	13	868732658834	30	15
393	14	868732661017	14	868732660947	70	35
394	15	868732663150	15	868732663060	90	45
395	16	868732665203	16	868732665183	20	10
396	17	868732667336	17	868732667296	40	20
397	18	868732669459	18	868732669409	50	25
398	19	868732671572	19	868732671522	50	25
399	20	868732673715	20	868732673645	70	35

400	21	868732675849	21	868732675758	91	45.5
401	22	868732677892	22	868732677871	21	10.5
402	23	868732680035	23	868732679985	50	25
403	24	868732682158	24	868732682108	50	25
404	25	868732684281	25	868732684221	60	30
405	26	868732686434	26	868732686334	100	50
406	27	868732688467	27	868732688447	20	10
407	28	868732690620	28	868732690560	60	30
408	29	868732692773	29	868732692683	90	45
409	30	868732694826	30	868732694796	30	15
410	31	868732696979	31	868732696909	70	35
411	32	868732699132	32	868732699042	90	45
412	33	868732701195	33	868732701155	40	20
413	34	868732703358	34	868732703268	90	45
414	35	868732705481	35	868732705391	90	45
415	36	868732707534	36	868732707504	30	15
416	37	868732709677	37	868732709617	60	30
417	38	868732711790	38	868732711740	50	25
418	39	868732713913	39	868732713853	60	30
419	40	868732716036	40	868732715966	70	35
420	41	868732718159	41	868732718079	80	40
421	42	868732720272	42	868732720202	70	35
422	43	868732722426	43	868732722315	111	55.5
423	44	868732724559	44	868732724458	101	50.5
424	45	868732726612	45	868732726582	30	15
425	46	868732728765	46	868732728695	70	35
426	47	868732730898	47	868732730808	90	45
427	48	868732732951	48	868732732921	30	15
428	49	868732735124	49	868732735044	80	40
429	50	868732737177	50	868732737157	20	10
430	51	868732739340	51	868732739270	70	35
431	52	868732741473	52	868732741383	90	45
432	53	868732743596	53	868732743506	90	45
433	54	868732745639	54	868732745619	20	10
434	55	868732747782	55	868732747732	50	25
435	56	868732749895	56	868732749845	50	25
436	57	868732752028	57	868732751958	70	35
437	58	868732754161	58	868732754081	80	40
438	59	868732756284	59	868732756194	90	45
439	60	868732758337	60	868732758307	30	15
440	61	868732760500	61	868732760430	70	35
441	62	868732762553	62	868732762543	10	5
442	63	868732764716	63	868732764656	60	30
443	64	868732766789	64	868732766779	10	5
444	65	868732768952	65	868732768892	60	30
445	66	868732771106	66	868732771015	91	45.5
446	67	868732773209	67	868732773128	81	40.5
447	68	868732775352	68	868732775262	90	45
448	69	868732777415	69	868732777375	40	20
449	70	868732779528	70	868732779488	40	20
450	71	868732781651	71	868732781601	50	25

451	72	868732783774	72	868732783724	50	25
452	73	868732785897	73	868732785837	60	30
453	74	868732788050	74	868732787960	90	45
454	75	868732790103	75	868732790073	30	15
455	76	868732792226	76	868732792186	40	20
456	77	868732794349	77	868732794299	50	25
457	78	868732796482	78	868732796412	70	35
458	79	868732798615	79	868732798535	80	40
459	80	868732800668	80	868732800648	20	10
460	81	868732802811	81	868732802761	50	25
461	82	868732804924	82	868732804874	50	25
462	83	868732807047	83	868732806997	50	25
463	84	868732809190	84	868732809110	80	40
464	85	868732811313	85	868732811223	90	45
465	86	868732813376	86	868732813346	30	15
466	87	868732815549	87	868732815459	90	45
467	88	868732817602	88	868732817572	30	15
468	89	868732819755	89	868732819685	70	35
469	90	868732821879	90	868732821798	81	40.5
470	91	868732824022	91	868732823921	101	50.5
471	92	868732826075	92	868732826035	40	20
472	93	868732828198	93	868732828148	50	25
473	94	868732830321	94	868732830261	60	30
474	95	868732832444	95	868732832384	60	30
475	96	868732834607	96	868732834507	100	50
476	97	868732836660	97	868732836620	40	20
477	98	868732838813	98	868732838733	80	40
478	99	868732840876	99	868732840856	20	10
479	100	868732843029	100	868732842969	60	30
480	101	868732845172	101	868732845082	90	45
481	102	868732847285	102	868732847195	90	45
482	103	868732849438	103	868732849338	100	50
483	104	868732851481	104	868732851451	30	15
484	105	868732853604	105	868732853564	40	20
485	106	868732855747	106	868732855687	60	30
486	107	868732857880	107	868732857800	80	40
487	108	868732860003	108	868732859913	90	45
488	109	868732862136	109	868732862046	90	45
489	110	868732864209	110	868732864159	50	25
490	111	868732866363	111	868732866272	91	45.5
491	112	868732868415	112	868732868385	30	15
492	113	868732870589	113	868732870508	81	40.5
493	114	868732872642	114	868732872622	20	10
494	115	868732874795	115	868732874735	60	30
495	116	868732876968	116	868732876878	90	45
496	117	868732879021	117	868732878991	30	15
497	118	868732881184	118	868732881104	80	40
498	119	868732883327	119	868732883237	90	45
499	120	868732885380	120	868732885350	30	15
500	121	868732887543	121	868732887473	70	35
501	122	868732889656	122	868732889586	70	35

502	123	868732891779	123	868732891699	80	40
503	124	868732893902	124	868732893822	80	40
504	125	868732895965	125	868732895945	20	10
505	0	868732898118	0	868732898058	60	30
506	1	868732900291	1	868732900191	100	50
507	2	868732902334	2	868732902304	30	15
508	3	868732904497	3	868732904417	80	40
509	4	868732906640	4	868732906540	100	50
510	5	868732908753	5	868732908663	90	45
511	6	868732910917	6	868732910816	101	50.5
512	7	868732912950	7	868732912929	21	10.5
513	8	868732915073	8	868732915043	30	15
514	9	868732917216	9	868732917166	50	25
515	10	868732919359	10	868732919279	80	40
516	11	868732921472	11	868732921392	80	40
517	12	868732923605	12	868732923515	90	45
518	13	868732925738	13	868732925628	110	55
519	14	868732927761	14	868732927741	20	10
520	15	868732929904	15	868732929864	40	20
521	16	868732932047	16	868732931977	70	35
522	17	868732934210	17	868732934110	100	50
523	18	868732936263	18	868732936223	40	20
524	19	868732938436	19	868732938346	90	45
525	20	868732940589	20	868732940489	100	50
526	21	868732942652	21	868732942602	50	25
527	22	868732944815	22	868732944715	100	50
528	23	868732946958	23	868732946868	90	45
529	24	868732949001	24	868732948981	20	10
530	25	868732951124	25	868732951094	30	15
531	26	868732953237	26	868732953217	20	10
532	27	868732955370	27	868732955330	40	20
533	28	868732957494	28	868732957443	51	25.5
534	29	868732959717	29	868732959627	90	45
535	30	868732961750	30	868732961740	10	5
536	31	868732963873	31	868732963853	20	10
537	32	868732966006	32	868732965976	30	15
538	33	868732968139	33	868732968089	50	25
539	34	868732970272	34	868732970202	70	35
540	35	868732972325	35	868732972315	10	5
541	36	868732974468	36	868732974428	40	20
542	37	868732976621	37	868732976551	70	35
543	38	868732978674	38	868732978664	10	5
544	39	868732980797	39	868732980777	20	10
545	40	868732982920	40	868732982890	30	15
546	41	868732985043	41	868732985013	30	15
547	42	868732987166	42	868732987126	40	20
548	43	868732989289	43	868732989239	50	25
549	44	868732991462	44	868732991352	110	55
550	45	868732993505	45	868732993475	30	15
551	46	868732995668	46	868732995588	80	40
552	47	868732997832	47	868732997731	101	50.5

553	48	868732999885	48	868732999854	31	15.5
554	49	868733002038	49	868733001968	70	35
555	50	868733004201	50	868733004101	100	50
556	51	868733006254	51	868733006214	40	20
557	52	868733008407	52	868733008337	70	35
558	53	868733010560	53	868733010460	100	50
559	54	868733012603	54	868733012573	30	15
560	55	868733014766	55	868733014696	70	35
561	56	868733016889	56	868733016809	80	40
562	57	868733019012	57	868733018922	90	45
563	58	868733021135	58	868733021045	90	45
564	59	868733023248	59	868733023158	90	45
565	60	868733025371	60	868733025281	90	45
566	61	868733027504	61	868733027404	100	50
567	62	868733029567	62	868733029517	50	25
568	63	868733031720	63	868733031630	90	45
569	64	868733033863	64	868733033773	90	45
570	65	868733035976	65	868733035886	90	45
571	66	868733038019	66	868733037999	20	10
572	67	868733040172	67	868733040122	50	25
573	68	868733042326	68	868733042235	91	45.5
574	69	868733044389	69	868733044348	41	20.5
575	70	868733046512	70	868733046461	51	25.5
576	71	868733048635	71	868733048585	50	25
577	72	868733050768	72	868733050698	70	35
578	73	868733052891	73	868733052811	80	40
579	74	868733055014	74	868733054924	90	45
580	75	868733057137	75	868733057047	90	45
581	76	868733059180	76	868733059160	20	10
582	77	868733061343	77	868733061273	70	35
583	78	868733063496	78	868733063396	100	50
584	79	868733065619	79	868733065519	100	50
585	80	868733067732	80	868733067642	90	45
586	81	868733069885	81	868733069765	120	60
587	82	868733072219	82	868733072148	71	35.5
588	83	868733074382	83	868733074281	101	50.5
589	84	868733076435	84	868733076405	30	15
590	85	868733078548	85	868733078518	30	15
591	86	868733080671	86	868733080631	40	20
592	87	868733082794	87	868733082744	50	25
593	88	868733084917	88	868733084867	50	25
594	89	868733087030	89	868733086980	50	25
595	90	868733089163	90	868733089093	70	35
596	91	868733091326	91	868733091226	100	50
597	92	868733093379	92	868733093349	30	15
598	93	868733095542	93	868733095462	80	40
599	94	868733097665	94	868733097575	90	45
600	95	868733099728	95	868733099688	40	20
601	96	868733101871	96	868733101811	60	30
602	97	868733103994	97	868733103924	70	35
603	98	868733106097	98	868733106037	60	30

604	99	868733108240	99	868733108160	80	40
605	100	868733110293	100	868733110273	20	10
606	101	868733112456	101	868733112396	60	30
607	102	868733114579	102	868733114509	70	35
608	103	868733116702	103	868733116622	80	40
609	104	868733118765	104	868733118735	30	15
610	105	868733120929	105	868733120858	71	35.5
611	106	868733123082	106	868733122982	100	50
612	107	868733125145	107	868733125105	40	20
613	108	868733127258	108	868733127218	40	20
614	109	868733129401	109	868733129331	70	35
615	110	868733131474	110	868733131444	30	15
616	111	868733133627	111	868733133567	60	30
617	112	868733135690	112	868733135680	10	5
618	113	868733137843	113	868733137793	50	25
619	114	868733140006	114	868733139916	90	45
620	115	868733142059	115	868733142029	30	15
621	116	868733144202	116	868733144142	60	30
622	117	868733146325	117	868733146255	70	35
623	118	868733148448	118	868733148368	80	40
624	119	868733150621	119	868733150521	100	50
625	120	868733152674	120	868733152634	40	20

APPENDIX P. EXPERIMENT 1 DATA FOR UDP

Experiment 1 (Server Control using UDP, Windows NT and Internet Explorer)

N	stat_u.txt Message Counter	stat_u.txt	sent_u.txt Message Counter	sent_u..txt	Round Trip Latency Time	Calculated one way latency time	Freq. Bin
1	0	868734767777	0	868734767626	151	75.5	0
2	1	868734769830	1	868734769810	20	10	6
3	2	868734771993	2	868734771923	70	35	11
4	3	868734774146	3	868734774046	100	50	16
5	4	868734776199	4	868734776159	40	20	21
6	5	868734778362	5	868734778282	80	40	26
7	6	868734780435	6	868734780395	40	20	31
8	7	868734782598	7	868734782508	90	45	36
9	8	868734784661	8	868734784621	40	20	41
10	9	868734786814	9	868734786744	70	35	46
11	10	868734788977	10	868734788877	100	50	51
12	11	868734791020	11	868734790990	30	15	56
13	12	868734793183	12	868734793113	70	35	61
14	13	868734795336	13	868734795246	90	45	66
15	14	868734797389	14	868734797359	30	15	71
16	15	868734799542	15	868734799472	70	35	76
17	16	868734801695	16	868734801605	90	45	81
18	17	868734803768	17	868734803718	50	25	
19	18	868734805931	18	868734805831	100	50	
20	19	868734807984	19	868734807944	40	20	
21	20	868734810128	20	868734810067	61	30.5	
22	21	868734812291	21	868734812200	91	45.5	
23	22	868734814354	22	868734814324	30	15	
24	23	868734816477	23	868734816437	40	20	
25	24	868734818600	24	868734818550	50	25	
26	25	868734820723	25	868734820663	60	30	
27	26	868734822876	26	868734822786	90	45	
28	27	868734825019	27	868734824929	90	45	
29	28	868734827072	28	868734827042	30	15	
30	29	868734829205	29	868734829155	50	25	
31	30	868734831328	30	868734831268	60	30	
32	31	868734833421	31	868734833391	30	15	
33	32	868734835564	32	868734835504	60	30	
34	33	868734837707	33	868734837617	90	45	
35	34	868734839760	34	868734839740	20	10	
36	35	868734841923	35	868734841853	70	35	
37	36	868734843986	36	868734843966	20	10	
38	37	868734846149	37	868734846089	60	30	
39	38	868734848272	38	868734848202	70	35	
40	39	868734850335	39	868734850315	20	10	
41	40	868734852508	40	868734852428	80	40	
42	41	868734854611	41	868734854551	60	30	
43	42	868734856765	42	868734856674	91	45.5	
44	43	868734858918	43	868734858838	80	40	
45	44	868734861041	44	868734860951	90	45	

46	45	868734863154	45	868734863074	80	40
47	46	868734865287	46	868734865207	80	40
48	47	868734867410	47	868734867320	90	45
49	48	868734869523	48	868734869443	80	40
50	49	868734871646	49	868734871566	80	40
51	50	868734873769	50	868734873689	80	40
52	51	868734875892	51	868734875802	90	45
53	52	868734878005	52	868734877925	80	40
54	53	868734880128	53	868734880038	90	45
55	54	868734882181	54	868734882151	30	15
56	55	868734884314	55	868734884274	40	20
57	56	868734886437	56	868734886387	50	25
58	57	868734888580	57	868734888500	80	40
59	58	868734890713	58	868734890623	90	45
60	59	868734892826	59	868734892746	80	40
61	60	868734894970	60	868734894879	91	45.5
62	61	868734897022	61	868734896992	30	15
63	62	868734899186	62	868734899105	81	40.5
64	63	868734901329	63	868734901239	90	45
65	64	868734903402	64	868734903352	50	25
66	65	868734905555	65	868734905475	80	40
67	66	868734907618	66	868734907588	30	15
68	67	868734909771	67	868734909701	70	35
69	68	868734911934	68	868734911834	100	50
70	69	868734913987	69	868734913957	30	15
71	70	868734916140	70	868734916070	70	35
72	71	868734918293	71	868734918203	90	45
73	72	868734920436	72	868734920316	120	60
74	73	868734922449	73	868734922429	20	10
75	74	868734924572	74	868734924542	30	15
76	75	868734926725	75	868734926665	60	30
77	76	868734928888	76	868734928798	90	45
78	77	868734930941	77	868734930911	30	15
79	78	868734933064	78	868734933024	40	20
80	79	868734935187	79	868734935147	40	20
81	80	868734937320	80	868734937260	60	30
82	81	868734939453	81	868734939373	80	40
83	82	868734941567	82	868734941486	81	40.5
84	83	868734943690	83	868734943609	81	40.5
85	84	868734945813	84	868734945722	91	45.5
86	85	868734947936	85	868734947846	90	45
87	86	868734950069	86	868734949979	90	45
88	87	868734952122	87	868734952102	20	10
89	88	868734954285	88	868734954215	70	35
90	89	868734956438	89	868734956348	90	45
91	90	868734958491	90	868734958461	30	15
92	91	868734960654	91	868734960584	70	35
93	92	868734962807	92	868734962727	80	40
94	93	868734964860	93	868734964840	20	10
95	94	868734967013	94	868734966953	60	30
96	95	868734969146	95	868734969066	80	40

97	96	868734971269	96	868734971189	80	40
98	97	868734973332	97	868734973302	30	15
99	98	868734975495	98	868734975415	80	40
100	99	868734977548	99	868734977538	10	5
101	100	868734979721	100	868734979651	70	35
102	101	868734981874	101	868734981784	90	45
103	102	868734983937	102	868734983897	40	20
104	103	868734986101	103	868734986020	81	40.5
105	104	868734988164	104	868734988133	31	15.5
106	105	868734990327	105	868734990247	80	40
107	106	868734992490	106	868734992400	90	45
108	107	868734994613	107	868734994513	100	50
109	108	868734996666	108	868734996636	30	15
110	109	868734998809	109	868734998749	60	30
111	110	868735000932	110	868735000862	70	35
112	111	868735003045	111	868735002985	60	30
113	112	868735005168	112	868735005098	70	35
114	113	868735007281	113	868735007211	70	35
115	114	868735009414	114	868735009334	80	40
116	115	868735011467	115	868735011447	20	10
117	116	868735013580	116	868735013560	20	10
118	117	868735015733	117	868735015683	50	25
119	118	868735017866	118	868735017796	70	35
120	119	868735019979	119	868735019909	70	35
121	120	868735022132	120	868735022042	90	45
122	121	868735024165	121	868735024155	10	5
123	122	868735026288	122	868735026268	20	10
124	123	868735028441	123	868735028391	50	25
125	124	868735030595	124	868735030504	91	45.5
126	125	868735032647	125	868735032617	30	15
127	0	868735034801	0	868735034730	71	35.5
128	1	868735036864	1	868735036854	10	5
129	2	868735039017	2	868735038967	50	25
130	3	868735041180	3	868735041090	90	45
131	4	868735043223	4	868735043203	20	10
132	5	868735045386	5	868735045316	70	35
133	6	868735047539	6	868735047449	90	45
134	7	868735049592	7	868735049562	30	15
135	8	868735051745	8	868735051685	60	30
136	9	868735053898	9	868735053808	90	45
137	10	868735055951	10	868735055921	30	15
138	11	868735058114	11	868735058034	80	40
139	12	868735060167	12	868735060157	10	5
140	13	868735062330	13	868735062270	60	30
141	14	868735064483	14	868735064393	90	45
142	15	868735066606	15	868735066526	80	40
143	16	868735068729	16	868735068639	90	45
144	17	868735070872	17	868735070772	100	50
145	18	868735072975	18	868735072895	80	40
146	19	868735075099	19	868735075018	81	40.5
147	20	868735077232	20	868735077141	91	45.5

148	21	868735079275	21	868735079254	21	10.5
149	22	868735081438	22	868735081378	60	30
150	23	868735083591	23	868735083521	70	35
151	24	868735085704	24	868735085634	70	35
152	25	868735087827	25	868735087747	80	40
153	26	868735089970	26	868735089870	100	50
154	27	868735092083	27	868735091993	90	45
155	28	868735094206	28	868735094116	90	45
156	29	868735096329	29	868735096239	90	45
157	30	868735098392	30	868735098352	40	20
158	31	868735100545	31	868735100465	80	40
159	32	868735102618	32	868735102578	40	20
160	33	868735104771	33	868735104701	70	35
161	34	868735106934	34	868735106834	100	50
162	35	868735108957	35	868735108947	10	5
163	36	868735111080	36	868735111060	20	10
164	37	868735113243	37	868735113173	70	35
165	38	868735115386	38	868735115306	80	40
166	39	868735117540	39	868735117449	91	45.5
167	40	868735119603	40	868735119562	41	20.5
168	41	868735121756	41	868735121675	81	40.5
169	42	868735123819	42	868735123799	20	10
170	43	868735125992	43	868735125912	80	40
171	44	868735128145	44	868735128055	90	45
172	45	868735130198	45	868735130168	30	15
173	46	868735132361	46	868735132281	80	40
174	47	868735134474	47	868735134404	70	35
175	48	868735136617	48	868735136527	90	45
176	49	868735138760	49	868735138670	90	45
177	50	868735140813	50	868735140783	30	15
178	51	868735142976	51	868735142906	70	35
179	52	868735145129	52	868735145029	100	50
180	53	868735147242	53	868735147152	90	45
181	54	868735149365	54	868735149265	100	50
182	55	868735151508	55	868735151408	100	50
183	56	868735153621	56	868735153531	90	45
184	57	868735155734	57	868735155644	90	45
185	58	868735157787	58	868735157767	20	10
186	59	868735159951	59	868735159880	71	35.5
187	60	868735162094	60	868735161993	101	50.5
188	61	868735164227	61	868735164117	110	55
189	62	868735166250	62	868735166230	20	10
190	63	868735168403	63	868735168343	60	30
191	64	868735170526	64	868735170466	60	30
192	65	868735172649	65	868735172579	70	35
193	66	868735174802	66	868735174712	90	45
194	67	868735176925	67	868735176835	90	45
195	68	868735179048	68	868735178948	100	50
196	69	868735181111	69	868735181071	40	20
197	70	868735183254	70	868735183184	70	35
198	71	868735185417	71	868735185317	100	50

199	72	868735187480	72	868735187440	40	20
200	73	868735189633	73	868735189553	80	40
201	74	868735191786	74	868735191696	90	45
202	75	868735193859	75	868735193809	50	25
203	76	868735196012	76	868735195922	90	45
204	77	868735198175	77	868735198075	100	50
205	78	868735200208	78	868735200188	20	10
206	79	868735202331	79	868735202301	30	15
207	80	868735204485	80	868735204424	61	30.5
208	81	868735206608	81	868735206538	70	35
209	82	868735208731	82	868735208651	80	40
210	83	868735210784	83	868735210774	10	5
211	84	868735212917	84	868735212887	30	15
212	85	868735215040	85	868735215000	40	20
213	86	868735217183	86	868735217113	70	35
214	87	868735219316	87	868735219236	80	40
215	88	868735221439	88	868735221349	90	45
216	89	868735223482	89	868735223462	20	10
217	90	868735225645	90	868735225585	60	30
218	91	868735227788	91	868735227698	90	45
219	92	868735229861	92	868735229811	50	25
220	93	868735232014	93	868735231924	90	45
221	94	868735234077	94	868735234047	30	15
222	95	868735236240	95	868735236160	80	40
223	96	868735238293	96	868735238273	20	10
224	97	868735240456	97	868735240386	70	35
225	98	868735242599	98	868735242509	90	45
226	99	868735244712	99	868735244622	90	45
227	100	868735246855	100	868735246745	110	55
228	101	868735248918	101	868735248868	50	25
229	102	868735251072	102	868735250981	91	45.5
230	103	868735253125	103	868735253094	31	15.5
231	104	868735255278	104	868735255218	60	30
232	105	868735257421	105	868735257331	90	45
233	106	868735259464	106	868735259444	20	10
234	107	868735261617	107	868735261557	60	30
235	108	868735263780	108	868735263680	100	50
236	109	868735265833	109	868735265793	40	20
237	110	868735267956	110	868735267916	40	20
238	111	868735270099	111	868735270029	70	35
239	112	868735272162	112	868735272142	20	10
240	113	868735274325	113	868735274265	60	30
241	114	868735276478	114	868735276388	90	45
242	115	868735278531	115	868735278511	20	10
243	116	868735280654	116	868735280624	30	15
244	117	868735282797	117	868735282737	60	30
245	118	868735284940	118	868735284850	90	45
246	119	868735287063	119	868735286963	100	50
247	120	868735289196	120	868735289086	110	55
248	121	868735291239	121	868735291199	40	20
249	122	868735293402	122	868735293312	90	45

250	123	868735295445	123	868735295425	20	10
251	124	868735297588	124	868735297548	40	20
252	125	868735299701	125	868735299661	40	20
253	0	868735301845	0	868735301774	71	35.5
254	1	868735303978	1	868735303898	80	40
255	2	868735306101	2	868735306011	90	45
256	3	868735308144	3	868735308124	20	10
257	4	868735310307	4	868735310247	60	30
258	5	868735312450	5	868735312360	90	45
259	6	868735314623	6	868735314523	100	50
260	7	868735316686	7	868735316636	50	25
261	8	868735318839	8	868735318749	90	45
262	9	868735321012	9	868735320912	100	50
263	10	868735323065	10	868735323025	40	20
264	11	868735325218	11	868735325148	70	35
265	12	868735327351	12	868735327271	80	40
266	13	868735329474	13	868735329384	90	45
267	14	868735331537	14	868735331497	40	20
268	15	868735333650	15	868735333620	30	15
269	16	868735335773	16	868735335733	40	20
270	17	868735337926	17	868735337856	70	35
271	18	868735340049	18	868735339969	80	40
272	19	868735342173	19	868735342082	91	45.5
273	20	868735344215	20	868735344205	10	5
274	21	868735346369	21	868735346319	50	25
275	22	868735348532	22	868735348442	90	45
276	23	868735350585	23	868735350555	30	15
277	24	868735352708	24	868735352668	40	20
278	25	868735354831	25	868735354791	40	20
279	26	868735356974	26	868735356904	70	35
280	27	868735359087	27	868735359017	70	35
281	28	868735361210	28	868735361140	70	35
282	29	868735363263	29	868735363253	10	5
283	30	868735365426	30	868735365376	50	25
284	31	868735367589	31	868735367489	100	50
285	32	868735369742	32	868735369722	20	10
286	33	868735371905	33	868735371845	60	30
287	34	868735373968	34	868735373958	10	5
288	35	868735376101	35	868735376071	30	15
289	36	868735378214	36	868735378184	30	15
290	37	868735380367	37	868735380307	60	30
291	38	868735382531	38	868735382440	91	45.5
292	39	868735384584	39	868735384553	31	15.5
293	40	868735386857	40	868735386797	60	30
294	41	868735389020	41	868735388920	100	50
295	42	868735391083	42	868735391043	40	20

APPENDIX Q. EXPERIMENT 2 DATA FOR TCP

Experiment 2 (Client Control using TCP, Windows NT and Internet Explorer)

N	stat_u.txt Message Counter	stat_u.txt	sent_u.txt Message Counter	sent_u..txt	Round Trip Latency Time	Calculated one way Latency time	Freq. Bin
1	0	868738526882	0	868738526802	80	40	0
2	1	868738528945	1	868738528925	20	10	6
3	2	868738531068	2	868738531038	30	15	11
4	3	868738533181	3	868738533161	20	10	16
5	4	868738535304	4	868738535274	30	15	21
6	5	868738537417	5	868738537387	30	15	26
7	6	868738539540	6	868738539510	30	15	31
8	7	868738541653	7	868738541623	30	15	36
9	8	868738543766	8	868738543736	30	15	41
10	9	868738545879	9	868738545859	20	10	46
11	10	868738547992	10	868738547972	20	10	51
12	11	868738550115	11	868738550095	20	10	56
13	12	868738552228	12	868738552208	20	10	61
14	13	868738554351	13	868738554321	30	15	66
15	14	868738556474	14	868738556444	30	15	71
16	15	868738558588	15	868738558557	31	15.5	76
17	16	868738560701	16	868738560671	30	15	81
18	17	868738562814	17	868738562794	20	10	86
19	18	868738564937	18	868738564907	30	15	91
20	19	868738567050	19	868738567020	30	15	
21	20	868738569173	20	868738569143	30	15	
22	21	868738571276	21	868738571256	20	10	
23	22	868738573389	22	868738573379	10	5	
24	23	868738575592	23	868738575492	100	50	
25	24	868738577695	24	868738577605	90	45	
26	25	868738579818	25	868738579728	90	45	
27	26	868738581931	26	868738581841	90	45	
28	27	868738584044	27	868738583954	90	45	
29	28	868738586157	28	868738586077	80	40	
30	29	868738588220	29	868738588190	30	15	
31	30	868738590343	30	868738590303	40	20	
32	31	868738592506	31	868738592426	80	40	
33	32	868738594639	32	868738594539	100	50	
34	33	868738596752	33	868738596662	90	45	
35	34	868738598815	34	868738598775	40	20	
36	35	868738600928	35	868738600898	30	15	
37	36	868738603061	36	868738603011	50	25	
38	37	868738605215	37	868738605124	91	45.5	
39	38	868738607318	38	868738607237	81	40.5	
40	39	868738609441	39	868738609361	80	40	
41	40	868738611564	40	868738611474	90	45	
42	41	868738613677	41	868738613597	80	40	
43	42	868738615790	42	868738615710	80	40	

44	43	868738617893	43	868738617823	70	35
45	44	868738620006	44	868738619946	60	30
46	45	868738622159	45	868738622059	100	50
47	46	868738624192	46	868738624182	10	5
48	47	868738626395	47	868738626295	100	50
49	48	868738628498	48	868738628408	90	45
50	49	868738630611	49	868738630531	80	40
51	50	868738632724	50	868738632644	80	40
52	51	868738634857	51	868738634757	100	50
53	52	868738636970	52	868738636880	90	45
54	53	868738639093	53	868738638993	100	50
55	54	868738641196	54	868738641106	90	45
56	55	868738643309	55	868738643229	80	40
57	56	868738645432	56	868738645342	90	45
58	57	868738647565	57	868738647465	100	50
59	58	868738649679	58	868738649588	91	45.5
60	59	868738651802	59	868738651701	101	50.5
61	60	868738653875	60	868738653814	61	30.5
62	61	868738655998	61	868738655938	60	30
63	62	868738658111	62	868738658051	60	30
64	63	868738660224	63	868738660174	50	25
65	64	868738662327	64	868738662287	40	20
66	65	868738664460	65	868738664400	60	30
67	66	868738666573	66	868738666523	50	25
68	67	868738668686	67	868738668636	50	25
69	68	868738670859	68	868738670749	110	55
70	69	868738672882	69	868738672872	10	5
71	70	868738675005	70	868738674985	20	10
72	71	868738677218	71	868738677098	120	60
73	72	868738679311	72	868738679221	90	45
74	73	868738681434	73	868738681334	100	50
75	74	868738683517	74	868738683457	60	30
76	75	868738685620	75	868738685570	50	25
77	76	868738687733	76	868738687683	50	25
78	77	868738689886	77	868738689806	80	40
79	78	868738691989	78	868738691919	70	35
80	79	868738694102	79	868738694032	70	35
81	80	868738696205	80	868738696155	50	25
82	81	868738698318	81	868738698278	40	20
83	82	868738700431	82	868738700391	40	20
84	83	868738702575	83	868738702504	71	35.5
85	84	868738704688	84	868738704618	70	35
86	85	868738706801	85	868738706741	60	30
87	86	868738708904	86	868738708854	50	25
88	87	868738711007	87	868738710977	30	15
89	88	868738713130	88	868738713090	40	20
90	89	868738715233	89	868738715203	30	15
91	90	868738717346	90	868738717326	20	10
92	91	868738719469	91	868738719439	30	15
93	92	868738721582	92	868738721562	20	10
94	93	868738723715	93	868738723675	40	20

95	94	868738725828	94	868738725788	40	20
96	95	868738727941	95	868738727911	30	15
97	96	868738730104	96	868738730024	80	40
98	97	868738732227	97	868738732137	90	45
99	98	868738734350	98	868738734260	90	45
100	99	868738736473	99	868738736373	100	50
101	100	868738738576	100	868738738486	90	45
102	101	868738740699	101	868738740609	90	45
103	102	868738742802	102	868738742722	80	40
104	103	868738744925	103	868738744845	80	40
105	104	868738747039	104	868738746958	81	40.5
106	105	868738749152	105	868738749071	81	40.5
107	106	868738751255	106	868738751184	71	35.5
108	107	868738753378	107	868738753308	70	35
109	108	868738755451	108	868738755421	30	15
110	109	868738757574	109	868738757534	40	20
111	110	868738759687	110	868738759657	30	15
112	111	868738761790	111	868738761770	20	10
113	112	868738763933	112	868738763893	40	20
114	113	868738766096	113	868738766006	90	45
115	114	868738768199	114	868738768119	80	40
116	115	868738770312	115	868738770242	70	35
117	116	868738772425	116	868738772355	70	35
118	117	868738774528	117	868738774468	60	30
119	118	868738776631	118	868738776591	40	20
120	119	868738778754	119	868738778704	50	25
121	120	868738780857	120	868738780817	40	20
122	121	868738782990	121	868738782940	50	25
123	122	868738785153	122	868738785053	100	50
124	123	868738787266	123	868738787176	90	45
125	124	868738789399	124	868738789289	110	55
126	125	868738791502	125	868738791412	90	45
127	0	868738793625	0	868738793525	100	50
128	1	868738795739	1	868738795648	91	45.5
129	2	868738797842	2	868738797761	81	40.5
130	3	868738799955	3	868738799874	81	40.5
131	4	868738802088	4	868738801998	90	45
132	5	868738804191	5	868738804111	80	40
133	6	868738806304	6	868738806224	80	40
134	7	868738808457	7	868738808347	110	55
135	8	868738810490	8	868738810460	30	15
136	9	868738812603	9	868738812583	20	10
137	10	868738814706	10	868738814696	10	5
138	11	868738816829	11	868738816809	20	10
139	12	868738819032	12	868738818922	110	55
140	13	868738821135	13	868738821045	90	45
141	14	868738823238	14	868738823158	80	40
142	15	868738825341	15	868738825271	70	35
143	16	868738827464	16	868738827394	70	35
144	17	868738829567	17	868738829507	60	30
145	18	868738831680	18	868738831630	50	25

146	19	868738833793	19	868738833743	50	25
147	20	868738835906	20	868738835856	50	25
148	21	868738838039	21	868738837979	60	30
149	22	868738840122	22	868738840092	30	15
150	23	868738842235	23	868738842205	30	15
151	24	868738844399	24	868738844328	71	35.5
152	25	868738846512	25	868738846441	71	35.5
153	26	868738848625	26	868738848554	71	35.5
154	27	868738850748	27	868738850678	70	35
155	28	868738852861	28	868738852791	70	35
156	29	868738854994	29	868738854904	90	45
157	30	868738857107	30	868738857027	80	40
158	31	868738859220	31	868738859140	80	40
159	32	868738861333	32	868738861253	80	40
160	33	868738863406	33	868738863376	30	15
161	34	868738865529	34	868738865489	40	20
162	35	868738867642	35	868738867602	40	20
163	36	868738869735	36	868738869725	10	5
164	37	868738871948	37	868738871838	110	55
165	38	868738874051	38	868738873951	100	50
166	39	868738876164	39	868738876074	90	45
167	40	868738878277	40	868738878187	90	45
168	41	868738880330	41	868738880310	20	10
169	42	868738882443	42	868738882423	20	10
170	43	868738884556	43	868738884536	20	10
171	44	868738886759	44	868738886659	100	50
172	45	868738888862	45	868738888772	90	45
173	46	868738890975	46	868738890885	90	45
174	47	868738893109	47	868738893008	101	50.5
175	48	868738895222	48	868738895121	101	50.5
176	49	868738897345	49	868738897244	101	50.5
177	50	868738899408	50	868738899358	50	25
178	51	868738901541	51	868738901471	70	35
179	52	868738903664	52	868738903594	70	35
180	53	868738905767	53	868738905707	60	30
181	54	868738907890	54	868738907820	70	35
182	55	868738910003	55	868738909943	60	30
183	56	868738912126	56	868738912056	70	35
184	57	868738914229	57	868738914179	50	25
185	58	868738916342	58	868738916292	50	25
186	59	868738918445	59	868738918405	40	20
187	60	868738920558	60	868738920518	40	20
188	61	868738922731	61	868738922641	90	45
189	62	868738924854	62	868738924754	100	50
190	63	868738926977	63	868738926877	100	50
191	64	868738929090	64	868738928990	100	50
192	65	868738931203	65	868738931113	90	45
193	66	868738933326	66	868738933226	100	50
194	67	868738935429	67	868738935349	80	40
195	68	868738937532	68	868738937462	70	35
196	69	868738939655	69	868738939575	80	40

197	70	868738941759	70	868738941698	61	30.5
198	71	868738943872	71	868738943811	61	30.5
199	72	868738945985	72	868738945924	61	30.5
200	73	868738948098	73	868738948048	50	25
201	74	868738950231	74	868738950161	70	35
202	75	868738952304	75	868738952274	30	15
203	76	868738954417	76	868738954397	20	10
204	77	868738956540	77	868738956510	30	15
205	78	868738958663	78	868738958623	40	20
206	79	868738960786	79	868738960746	40	20
207	80	868738962909	80	868738962859	50	25
208	81	868738965012	81	868738964972	40	20
209	82	868738967115	82	868738967095	20	10
210	83	868738969238	83	868738969218	20	10
211	84	868738971341	84	868738971331	10	5
212	85	868738973554	85	868738973444	110	55
213	86	868738975617	86	868738975567	50	25
214	87	868738977730	87	868738977680	50	25
215	88	868738979843	88	868738979793	50	25
216	89	868738981996	89	868738981916	80	40
217	90	868738984119	90	868738984029	90	45
218	91	868738986232	91	868738986152	80	40
219	92	868738988345	92	868738988265	80	40
220	93	868738990449	93	868738990378	71	35.5
221	94	868738992552	94	868738992491	61	30.5
222	95	868738994675	95	868738994615	60	30
223	96	868738996788	96	868738996728	60	30
224	97	868738998911	97	868738998851	60	30
225	98	868739001024	98	868739000964	60	30
226	99	868739003127	99	868739003077	50	25
227	100	868739005270	100	868739005200	70	35
228	101	868739007403	101	868739007313	90	45
229	102	868739009536	102	868739009426	110	55
230	103	868739011639	103	868739011549	90	45
231	104	868739013742	104	868739013662	80	40
232	105	868739015865	105	868739015775	90	45
233	106	868739017968	106	868739017898	70	35
234	107	868739020081	107	868739020011	70	35
235	108	868739022184	108	868739022134	50	25
236	109	868739024357	109	868739024247	110	55
237	110	868739026460	110	868739026360	100	50
238	111	868739028583	111	868739028483	100	50
239	112	868739030686	112	868739030596	90	45
240	113	868739032819	113	868739032719	100	50
241	114	868739034953	114	868739034832	121	60.5
242	115	868739037066	115	868739036945	121	60.5
243	116	868739039179	116	868739039068	111	55.5
244	117	868739041212	117	868739041181	31	15.5
245	118	868739043315	118	868739043295	20	10
246	119	868739045438	119	868739045418	20	10
247	120	868739047591	120	868739047531	60	30

248	121	868739049724	121	868739049654	70	35
249	122	868739051847	122	868739051767	80	40
250	123	868739053960	123	868739053880	80	40
251	124	868739056073	124	868739056003	70	35
252	125	868739058196	125	868739058116	80	40
253	0	868739060309	0	868739060229	80	40
254	1	868739062422	1	868739062352	70	35
255	2	868739064545	2	868739064465	80	40
256	3	868739066688	3	868739066578	110	55
257	4	868739068801	4	868739068701	100	50
258	5	868739070904	5	868739070814	90	45
259	6	868739073017	6	868739072937	80	40
260	7	868739075150	7	868739075050	100	50
261	8	868739077203	8	868739077173	30	15
262	9	868739079336	9	868739079286	50	25
263	10	868739081459	10	868739081409	50	25
264	11	868739083572	11	868739083522	50	25
265	12	868739085665	12	868739085635	30	15
266	13	868739087778	13	868739087758	20	10
267	14	868739089881	14	868739089871	10	5
268	15	868739092005	15	868739091985	20	10
269	16	868739094148	16	868739094108	40	20
270	17	868739096261	17	868739096221	40	20
271	18	868739098364	18	868739098334	30	15
272	19	868739100487	19	868739100457	30	15
273	20	868739102590	20	868739102570	20	10
274	21	868739104713	21	868739104683	30	15
275	22	868739106916	22	868739106806	110	55
276	23	868739109019	23	868739108919	100	50
277	24	868739111132	24	868739111032	100	50
278	25	868739113255	25	868739113155	100	50
279	26	868739115358	26	868739115268	90	45
280	27	868739117481	27	868739117391	90	45
281	28	868739119554	28	868739119504	50	25
282	29	868739121677	29	868739121617	60	30
283	30	868739123800	30	868739123740	60	30
284	31	868739125873	31	868739125853	20	10
285	32	868739127996	32	868739127966	30	15
286	33	868739130119	33	868739130089	30	15
287	34	868739132232	34	868739132202	30	15
288	35	868739134345	35	868739134315	30	15
289	36	868739136489	36	868739136438	51	25.5
290	37	868739138602	37	868739138551	51	25.5
291	38	868739140715	38	868739140665	50	25
292	39	868739142878	39	868739142788	90	45
293	40	868739145001	40	868739144931	70	35
294	41	868739147134	41	868739147044	90	45
295	42	868739149237	42	868739149157	80	40
296	43	868739151350	43	868739151280	70	35
297	44	868739153463	44	868739153403	60	30
298	45	868739155576	45	868739155516	60	30

299	46	868739157679	46	868739157629	50	25
300	47	868739159792	47	868739159752	40	20
301	48	868739161905	48	868739161865	40	20
302	49	868739164008	49	868739163978	30	15
303	50	868739166141	50	868739166101	40	20
304	51	868739168284	51	868739168214	70	35
305	52	868739170407	52	868739170327	80	40
306	53	868739172510	53	868739172450	60	30
307	54	868739174613	54	868739174563	50	25
308	55	868739176746	55	868739176676	70	35
309	56	868739178889	56	868739178799	90	45
310	57	868739181003	57	868739180912	91	45.5
311	58	868739183126	58	868739183025	101	50.5
312	59	868739185249	59	868739185148	101	50.5
313	60	868739187352	60	868739187262	90	45
314	61	868739189475	61	868739189385	90	45
315	62	868739191528	62	868739191498	30	15
316	63	868739193641	63	868739193611	30	15
317	64	868739195754	64	868739195734	20	10
318	65	868739197867	65	868739197847	20	10
319	66	868739199980	66	868739199960	20	10
320	67	868739202093	67	868739202083	10	5
321	68	868739204306	68	868739204196	110	55
322	69	868739206409	69	868739206319	90	45
323	70	868739208512	70	868739208432	80	40
324	71	868739210615	71	868739210555	60	30
325	72	868739212728	72	868739212668	60	30
326	73	868739214811	73	868739214781	30	15
327	74	868739216944	74	868739216904	40	20
328	75	868739219067	75	868739219017	50	25
329	76	868739221180	76	868739221130	50	25
330	77	868739223293	77	868739223253	40	20
331	78	868739225406	78	868739225376	30	15
332	79	868739227509	79	868739227489	20	10
333	80	868739229622	80	868739229602	20	10
334	81	868739231745	81	868739231725	20	10
335	82	868739233869	82	868739233838	31	15.5
336	83	868739235972	83	868739235952	20	10
337	84	868739238175	84	868739238075	100	50
338	85	868739240288	85	868739240188	100	50
339	86	868739242411	86	868739242311	100	50
340	87	868739244564	87	868739244424	140	70
341	88	868739246587	88	868739246537	50	25
342	89	868739248710	89	868739248660	50	25
343	90	868739250813	90	868739250773	40	20
344	91	868739252926	91	868739252886	40	20
345	92	868739255029	92	868739255009	20	10
346	93	868739257142	93	868739257122	20	10
347	94	868739259345	94	868739259245	100	50
348	95	868739261378	95	868739261358	20	10
349	96	868739263501	96	868739263481	20	10

350	97	868739265614	97	868739265594	20	10
351	98	868739267737	98	868739267707	30	15
352	99	868739269850	99	868739269830	20	10
353	100	868739271963	100	868739271943	20	10
354	101	868739274086	101	868739274056	30	15
355	102	868739276199	102	868739276179	20	10
356	103	868739278312	103	868739278292	20	10
357	104	868739280425	104	868739280405	20	10
358	105	868739282629	105	868739282529	100	50
359	106	868739284672	106	868739284652	20	10
360	107	868739286835	107	868739286765	70	35
361	108	868739288968	108	868739288878	90	45
362	109	868739291101	109	868739291001	100	50
363	110	868739293174	110	868739293114	60	30
364	111	868739295297	111	868739295227	70	35
365	112	868739297420	112	868739297350	70	35
366	113	868739299533	113	868739299463	70	35
367	114	868739301636	114	868739301576	60	30
368	115	868739303749	115	868739303699	50	25
369	116	868739305862	116	868739305812	50	25
370	117	868739307975	117	868739307935	40	20
371	118	868739310098	118	868739310048	50	25
372	119	868739312201	119	868739312171	30	15
373	120	868739314304	120	868739314284	20	10
374	121	868739316417	121	868739316397	20	10
375	122	868739318620	122	868739318520	100	50
376	123	868739320743	123	868739320633	110	55
377	124	868739322897	124	868739322746	151	75.5
378	125	868739324899	125	868739324869	30	15
379	0	868739327012	0	868739326982	30	15
380	1	868739329126	1	868739329095	31	15.5
381	2	868739331229	2	868739331219	10	5
382	3	868739333362	3	868739333332	30	15
383	4	868739335475	4	868739335445	30	15
384	5	868739337588	5	868739337568	20	10
385	6	868739339801	6	868739339681	120	60
386	7	868739341814	7	868739341804	10	5
387	8	868739344027	8	868739343917	110	55
388	9	868739346050	9	868739346030	20	10
389	10	868739348163	10	868739348143	20	10
390	11	868739350286	11	868739350266	20	10
391	12	868739352409	12	868739352389	20	10
392	13	868739354512	13	868739354502	10	5
393	14	868739356635	14	868739356615	20	10
394	15	868739358888	15	868739358738	150	75
395	16	868739360891	16	868739360851	40	20
396	17	868739363004	17	868739362964	40	20
397	18	868739365187	18	868739365087	100	50
398	19	868739367290	19	868739367200	90	45
399	20	868739369413	20	868739369323	90	45
400	21	868739371506	21	868739371436	70	35

401	22	868739373619	22	868739373549	70	35
402	23	868739375753	23	868739375662	91	45.5
403	24	868739377886	24	868739377785	101	50.5
404	25	868739379989	25	868739379899	90	45
405	26	868739382112	26	868739382022	90	45
406	27	868739384165	27	868739384135	30	15
407	28	868739386278	28	868739386248	30	15
408	29	868739388411	29	868739388371	40	20
409	30	868739390584	30	868739390484	100	50
410	31	868739392697	31	868739392597	100	50
411	32	868739394770	32	868739394720	50	25
412	33	868739396883	33	868739396833	50	25
413	34	868739399016	34	868739398946	70	35
414	35	868739401119	35	868739401069	50	25
415	36	868739403222	36	868739403182	40	20
416	37	868739405345	37	868739405305	40	20
417	38	868739407458	38	868739407418	40	20
418	39	868739409561	39	868739409531	30	15
419	40	868739411694	40	868739411654	40	20
420	41	868739413797	41	868739413767	30	15
421	42	868739415910	42	868739415890	20	10
422	43	868739418023	43	868739418003	20	10
423	44	868739420136	44	868739420116	20	10
424	45	868739422350	45	868739422239	111	55.5
425	46	868739424372	46	868739424352	20	10
426	47	868739426486	47	868739426465	21	10.5
427	48	868739428699	48	868739428589	110	55
428	49	868739430812	49	868739430702	110	55
429	50	868739432915	50	868739432815	100	50
430	51	868739435028	51	868739434938	90	45
431	52	868739437091	52	868739437051	40	20
432	53	868739439204	53	868739439164	40	20
433	54	868739441317	54	868739441287	30	15
434	55	868739443430	55	868739443400	30	15
435	56	868739445543	56	868739445513	30	15
436	57	868739447656	57	868739447636	20	10
437	58	868739449859	58	868739449749	110	55
438	59	868739451962	59	868739451872	90	45
439	60	868739454085	60	868739453985	100	50
440	61	868739456198	61	868739456108	90	45
441	62	868739458321	62	868739458221	100	50
442	63	868739460424	63	868739460334	90	45
443	64	868739462547	64	868739462457	90	45
444	65	868739464650	65	868739464570	80	40
445	66	868739466723	66	868739466683	40	20
446	67	868739468836	67	868739468806	30	15
447	68	868739470939	68	868739470919	20	10
448	69	868739473052	69	868739473032	20	10
449	70	868739475166	70	868739475155	11	5.5
450	71	868739477379	71	868739477269	110	55
451	72	868739479492	72	868739479382	110	55

452	73	868739481605	73	868739481505	100	50
453	74	868739483718	74	868739483618	100	50
454	75	868739485821	75	868739485741	80	40
455	76	868739487944	76	868739487854	90	45
456	77	868739490067	77	868739489967	100	50
457	78	868739492110	78	868739492090	20	10
458	79	868739494223	79	868739494203	20	10
459	80	868739496396	80	868739496316	80	40
460	81	868739498519	81	868739498439	80	40
461	82	868739500642	82	868739500552	90	45
462	83	868739502755	83	868739502665	90	45
463	84	868739504858	84	868739504788	70	35
464	85	868739506981	85	868739506901	80	40
465	86	868739509114	86	868739509024	90	45
466	87	868739511237	87	868739511137	100	50
467	88	868739513350	88	868739513260	90	45
468	89	868739515453	89	868739515373	80	40
469	90	868739517566	90	868739517486	80	40
470	91	868739519710	91	868739519609	101	50.5
471	92	868739521833	92	868739521722	111	55.5
472	93	868739523956	93	868739523846	110	55
473	94	868739526069	94	868739525959	110	55
474	95	868739528182	95	868739528082	100	50
475	96	868739530295	96	868739530205	90	45
476	97	868739532418	97	868739532318	100	50
477	98	868739534531	98	868739534431	100	50
478	99	868739536654	99	868739536554	100	50
479	100	868739538727	100	868739538667	60	30
480	101	868739540840	101	868739540780	60	30
481	102	868739542973	102	868739542903	70	35
482	103	868739545086	103	868739545016	70	35
483	104	868739547209	104	868739547129	80	40
484	105	868739549312	105	868739549252	60	30
485	106	868739551405	106	868739551365	40	20
486	107	868739553518	107	868739553488	30	15
487	108	868739555671	108	868739555601	70	35
488	109	868739557784	109	868739557714	70	35
489	110	868739559907	110	868739559837	70	35
490	111	868739561970	111	868739561950	20	10
491	112	868739564083	112	868739564063	20	10
492	113	868739566206	113	868739566186	20	10
493	114	868739568319	114	868739568299	20	10
494	115	868739570523	115	868739570412	111	55.5
495	116	868739572556	116	868739572536	20	10
496	117	868739574669	117	868739574649	20	10
497	118	868739576792	118	868739576762	30	15
498	119	868739578905	119	868739578885	20	10
499	120	868739581078	120	868739580998	80	40
500	121	868739583181	121	868739583121	60	30
501	122	868739585304	122	868739585234	70	35
502	123	868739587427	123	868739587347	80	40

503	124	868739589540	124	868739589470	70	35
504	125	868739591613	125	868739591583	30	15
505	0	868739593716	0	868739593696	20	10
506	1	868739595839	1	868739595819	20	10
507	2	868739597942	2	868739597932	10	5
508	3	868739600075	3	868739600045	30	15
509	4	868739602198	4	868739602168	30	15
510	5	868739604311	5	868739604281	30	15
511	6	868739606434	6	868739606394	40	20
512	7	868739608587	7	868739608517	70	35
513	8	868739610710	8	868739610630	80	40
514	9	868739612823	9	868739612743	80	40
515	10	868739614946	10	868739614866	80	40
516	11	868739617050	11	868739616979	71	35.5
517	12	868739619173	12	868739619102	71	35.5
518	13	868739621306	13	868739621216	90	45
519	14	868739623439	14	868739623339	100	50
520	15	868739625552	15	868739625452	100	50
521	16	868739627655	16	868739627575	80	40
522	17	868739629758	17	868739629688	70	35
523	18	868739631871	18	868739631801	70	35
524	19	868739633994	19	868739633924	70	35
525	20	868739636107	20	868739636037	70	35
526	21	868739638280	21	868739638150	130	65
527	22	868739640293	22	868739640273	20	10
528	23	868739642406	23	868739642386	20	10
529	24	868739644609	24	868739644499	110	55
530	25	868739646732	25	868739646622	110	55
531	26	868739648755	26	868739648735	20	10
532	27	868739650958	27	868739650858	100	50
533	28	868739653071	28	868739652971	100	50
534	29	868739655194	29	868739655094	100	50
535	30	868739657297	30	868739657207	90	45
536	31	868739659410	31	868739659320	90	45
537	32	868739661533	32	868739661443	90	45
538	33	868739663586	33	868739663556	30	15
539	34	868739665689	34	868739665669	20	10
540	35	868739667873	35	868739667792	81	40.5
541	36	868739670717	36	868739669906	81	40.5
542	37	868739672089	37	868739672019	70	35
543	38	868739674172	38	868739674142	30	15
544	39	868739676375	39	868739676255	120	60
545	40	868739678468	40	868739678368	100	50
546	41	868739680581	41	868739680491	90	45
547	42	868739682724	42	868739682604	120	60
548	43	868739684757	43	868739684727	30	15
549	44	868739686870	44	868739686840	30	15
550	45	868739688993	45	868739688953	40	20
551	46	868739691146	46	868739691076	70	35
552	47	868739693209	47	868739693189	20	10
553	48	868739695392	48	868739695302	90	45

554	49	868739697535	49	868739697425	110	55
555	50	868739699578	50	868739699538	40	20
556	51	868739701791	51	868739701661	130	65
557	52	868739703844	52	868739703774	70	35
558	53	868739705917	53	868739705887	30	15
559	54	868739708130	54	868739708010	120	60
560	55	868739710193	55	868739710123	70	35
561	56	868739712367	56	868739712246	121	60.5
562	57	868739714500	57	868739714359	141	70.5
563	58	868739716553	58	868739716483	70	35
564	59	868739718636	59	868739718596	40	20
565	60	868739720839	60	868739720709	130	65
566	61	868739722922	61	868739722832	90	45
567	62	868739725025	62	868739724945	80	40
568	63	868739727088	63	868739727058	30	15
569	64	868739729251	64	868739729181	70	35
570	65	868739731384	65	868739731294	90	45
571	66	868739733487	66	868739733407	80	40
572	67	868739735550	67	868739735530	20	10
573	68	868739737743	68	868739737643	100	50
574	69	868739739846	69	868739739756	90	45
575	70	868739741929	70	868739741879	50	25
576	71	868739744042	71	868739743992	50	25
577	72	868739746145	72	868739746105	40	20
578	73	868739748318	73	868739748228	90	45
579	74	868739750421	74	868739750341	80	40
580	75	868739752534	75	868739752454	80	40
581	76	868739754597	76	868739754577	20	10
582	77	868739756710	77	868739756690	20	10
583	78	868739758944	78	868739758803	141	70.5
584	79	868739761027	79	868739760926	101	50.5
585	80	868739763090	80	868739763039	51	25.5
586	81	868739765243	81	868739765162	81	40.5
587	82	868739767376	82	868739767276	100	50
588	83	868739769519	83	868739769389	130	65
589	84	868739771662	84	868739771542	120	60
590	85	868739773785	85	868739773655	130	65
591	86	868739775868	86	868739775768	100	50
592	87	868739777941	87	868739777891	50	25
593	88	868739780064	88	868739780004	60	30
594	89	868739782227	89	868739782117	110	55
595	90	868739784360	90	868739784240	120	60
596	91	868739786413	91	868739786353	60	30
597	92	868739788516	92	868739788466	50	25
598	93	868739790719	93	868739790589	130	65
599	94	868739792802	94	868739792712	90	45
600	95	868739795005	95	868739794825	180	90
601	96	868739797058	96	868739796938	120	60
602	97	868739799091	97	868739799061	30	15
603	98	868739801264	98	868739801184	80	40
604	99	868739803317	99	868739803297	20	10

605	100	868739805551	100	868739805410	141	70.5
606	101	868739807634	101	868739807533	101	50.5
607	102	868739809727	102	868739809646	81	40.5
608	103	868739811900	103	868739811760	140	70
609	104	868739813903	104	868739813883	20	10
610	105	868739816026	105	868739815996	30	15

APPENDIX R. EXPERIMENT 2 DATA FOR UDP

Experiment 2 (Server Control using UDP, Windows NT and Internet Explorer)

N	stat_u.txt Message Counter	stat_u.txt	sent_u.txt Message Counter	sent_u..txt	Round Trip Latency Time	Calculated one way Latency time	Freq. Bin
1	0	868736297196	0	868736297076	120	60	0
2	1	868736299239	1	868736299199	40	20	6
3	2	868736301342	2	868736301312	30	15	11
4	3	868736303465	3	868736303425	40	20	16
5	4	868736305568	4	868736305548	20	10	21
6	5	868736307681	5	868736307661	20	10	26
7	6	868736309814	6	868736309774	40	20	31
8	7	868736311917	7	868736311897	20	10	36
9	8	868736314030	8	868736314010	20	10	41
10	9	868736316153	9	868736316123	30	15	46
11	10	868736318266	10	868736318246	20	10	51
12	11	868736320379	11	868736320359	20	10	56
13	12	868736322492	12	868736322472	20	10	61
14	13	868736324615	13	868736324595	20	10	66
15	14	868736326728	14	868736326708	20	10	71
16	15	868736328851	15	868736328821	30	15	76
17	16	868736330954	16	868736330944	10	5	81
18	17	868736333077	17	868736333057	20	10	86
19	18	868736335281	18	868736335170	111	55.5	91
20	19	868736337384	19	868736337293	91	45.5	96
21	20	868736339497	20	868736339407	90	45	101
22	21	868736341610	21	868736341530	80	40	
23	22	868736343723	22	868736343643	80	40	
24	23	868736345836	23	868736345766	70	35	
25	24	868736347969	24	868736347879	90	45	
26	25	868736350072	25	868736349992	80	40	
27	26	868736352175	26	868736352115	60	30	
28	27	868736354288	27	868736354228	60	30	
29	28	868736356381	28	868736356341	40	20	
30	29	868736358484	29	868736358464	20	10	
31	30	868736360597	30	868736360577	20	10	
32	31	868736362780	31	868736362690	90	45	
33	32	868736364893	32	868736364813	80	40	
34	33	868736366996	33	868736366926	70	35	
35	34	868736369109	34	868736369039	70	35	
36	35	868736371232	35	868736371162	70	35	
37	36	868736373375	36	868736373275	100	50	
38	37	868736375488	37	868736375398	90	45	
39	38	868736377601	38	868736377511	90	45	
40	39	868736379704	39	868736379634	70	35	
41	40	868736381817	40	868736381747	70	35	
42	41	868736383961	41	868736383860	101	50.5	
43	42	868736386014	42	868736385983	31	15.5	
44	43	868736388187	43	868736388097	90	45	

45	44	868736390300	44	868736390210	90	45
46	45	868736392413	45	868736392333	80	40
47	46	868736394516	46	868736394456	60	30
48	47	868736396639	47	868736396569	70	35
49	48	868736398752	48	868736398692	60	30
50	49	868736400865	49	868736400805	60	30
51	50	868736402968	50	868736402918	50	25
52	51	868736405071	51	868736405041	30	15
53	52	868736407174	52	868736407154	20	10
54	53	868736409337	53	868736409267	70	35
55	54	868736411450	54	868736411390	60	30
56	55	868736413553	55	868736413503	50	25
57	56	868736415666	56	868736415616	50	25
58	57	868736417779	57	868736417739	40	20
59	58	868736419902	58	868736419852	50	25
60	59	868736422075	59	868736421975	100	50
61	60	868736424198	60	868736424088	110	55
62	61	868736426311	61	868736426211	100	50
63	62	868736428414	62	868736428324	90	45
64	63	868736430528	63	868736430437	91	45.5
65	64	868736432631	64	868736432560	71	35.5
66	65	868736434734	65	868736434673	61	30.5
67	66	868736436837	66	868736436797	40	20
68	67	868736438940	67	868736438910	30	15
69	68	868736441043	68	868736441023	20	10
70	69	868736443246	69	868736443146	100	50
71	70	868736445279	70	868736445259	20	10
72	71	868736447392	71	868736447372	20	10
73	72	868736449515	72	868736449495	20	10
74	73	868736451718	73	868736451608	110	55
75	74	868736453821	74	868736453731	90	45
76	75	868736455924	75	868736455844	80	40
77	76	868736458047	76	868736457967	80	40
78	77	868736460160	77	868736460080	80	40
79	78	868736462273	78	868736462193	80	40
80	79	868736464376	79	868736464316	60	30
81	80	868736466489	80	868736466429	60	30
82	81	868736468602	81	868736468542	60	30
83	82	868736470705	82	868736470665	40	20
84	83	868736472808	83	868736472778	30	15
85	84	868736474911	84	868736474901	10	5
86	85	868736477125	85	868736477014	111	55.5
87	86	868736479228	86	868736479137	91	45.5
88	87	868736481341	87	868736481250	91	45.5
89	88	868736483444	88	868736483374	70	35
90	89	868736485557	89	868736485487	70	35
91	90	868736487630	90	868736487600	30	15
92	91	868736489733	91	868736489723	10	5
93	92	868736491856	92	868736491836	20	10
94	93	868736494059	93	868736493949	110	55
95	94	868736496172	94	868736496072	100	50

96	95	868736498295	95	868736498185	110	55
97	96	868736500338	96	868736500308	30	15
98	97	868736502441	97	868736502421	20	10
99	98	868736504604	98	868736504534	70	35
100	99	868736506717	99	868736506657	60	30
101	100	868736508840	100	868736508770	70	35
102	101	868736510913	101	868736510883	30	15
103	102	868736513026	102	868736513006	20	10
104	103	868736515139	103	868736515119	20	10
105	104	868736517252	104	868736517242	10	5
106	105	868736519455	105	868736519355	100	50
107	106	868736521498	106	868736521468	30	15
108	107	868736523651	107	868736523591	60	30
109	108	868736525744	108	868736525704	40	20
110	109	868736527847	109	868736527817	30	15
111	110	868736529950	110	868736529940	10	5
112	111	868736532154	111	868736532054	100	50
113	112	868736534257	112	868736534177	80	40
114	113	868736536380	113	868736536290	90	45
115	114	868736538493	114	868736538403	90	45
116	115	868736540586	115	868736540526	60	30
117	116	868736542699	116	868736542639	60	30
118	117	868736544802	117	868736544752	50	25
119	118	868736546915	118	868736546875	40	20
120	119	868736549018	119	868736548988	30	15
121	120	868736551141	120	868736551101	40	20
122	121	868736553234	121	868736553224	10	5
123	122	868736555357	122	868736555337	20	10
124	123	868736557480	123	868736557460	20	10
125	124	868736559603	124	868736559573	30	15
126	125	868736561716	125	868736561686	30	15
127	0	868736563829	0	868736563809	20	10
128	1	868736566032	1	868736565922	110	55
129	2	868736568145	2	868736568045	100	50
130	3	868736570248	3	868736570158	90	45
131	4	868736572361	4	868736572271	90	45
132	5	868736574495	5	868736574394	101	50.5
133	6	868736576558	6	868736576507	51	25.5
134	7	868736578661	7	868736578630	31	15.5
135	8	868736580794	8	868736580744	50	25
136	9	868736582897	9	868736582857	40	20
137	10	868736585010	10	868736584980	30	15
138	11	868736587123	11	868736587093	30	15
139	12	868736589226	12	868736589206	20	10
140	13	868736591349	13	868736591329	20	10
141	14	868736593462	14	868736593442	20	10
142	15	868736595665	15	868736595555	110	55
143	16	868736597788	16	868736597678	110	55
144	17	868736599951	17	868736599791	160	80
145	18	868736601974	18	868736601914	60	30
146	19	868736604107	19	868736604027	80	40

147	20	868736606220	20	868736606140	80	40
148	21	868736608323	21	868736608263	60	30
149	22	868736610446	22	868736610376	70	35
150	23	868736612549	23	868736612489	60	30
151	24	868736614662	24	868736614612	50	25
152	25	868736616765	25	868736616725	40	20
153	26	868736618938	26	868736618848	90	45
154	27	868736621062	27	868736620961	101	50.5
155	28	868736623165	28	868736623074	91	45.5
156	29	868736625268	29	868736625197	71	35.5
157	30	868736627391	30	868736627310	81	40.5
158	31	868736629564	31	868736629434	130	65
159	32	868736631587	32	868736631547	40	20
160	33	868736633700	33	868736633660	40	20
161	34	868736635813	34	868736635783	30	15
162	35	868736637936	35	868736637896	40	20
163	36	868736640049	36	868736640009	40	20
164	37	868736642152	37	868736642132	20	10
165	38	868736644275	38	868736644245	30	15
166	39	868736646388	39	868736646358	30	15
167	40	868736648501	40	868736648481	20	10
168	41	868736650624	41	868736650604	20	10
169	42	868736652747	42	868736652717	30	15
170	43	868736654850	43	868736654830	20	10
171	44	868736657063	44	868736656953	110	55
172	45	868736659166	45	868736659066	100	50
173	46	868736661269	46	868736661189	80	40
174	47	868736663402	47	868736663302	100	50
175	48	868736665445	48	868736665415	30	15
176	49	868736667568	49	868736667538	30	15
177	50	868736669681	50	868736669651	30	15
178	51	868736671804	51	868736671764	40	20
179	52	868736673907	52	868736673887	20	10
180	53	868736676031	53	868736676001	30	15
181	54	868736678144	54	868736678114	30	15
182	55	868736680347	55	868736680237	110	55
183	56	868736682370	56	868736682350	20	10
184	57	868736684483	57	868736684473	10	5
185	58	868736686616	58	868736686586	30	15
186	59	868736688729	59	868736688699	30	15
187	60	868736690872	60	868736690822	50	25
188	61	868736692985	61	868736692935	50	25
189	62	868736695098	62	868736695048	50	25
190	63	868736697271	63	868736697171	100	50
191	64	868736699394	64	868736699294	100	50
192	65	868736701567	65	868736701407	160	80
193	66	868736703570	66	868736703520	50	25
194	67	868736705693	67	868736705643	50	25
195	68	868736707846	68	868736707756	90	45
196	69	868736709949	69	868736709879	70	35
197	70	868736712062	70	868736711992	70	35

198	71	868736714165	71	868736714105	60	30
199	72	868736716288	72	868736716228	60	30
200	73	868736718391	73	868736718341	50	25
201	74	868736720484	74	868736720454	30	15
202	75	868736722598	75	868736722577	21	10.5
203	76	868736724711	76	868736724691	20	10
204	77	868736726914	77	868736726804	110	55
205	78	868736729017	78	868736728927	90	45
206	79	868736731140	79	868736731040	100	50
207	80	868736733243	80	868736733163	80	40
208	81	868736735366	81	868736735276	90	45
209	82	868736737419	82	868736737389	30	15
210	83	868736739532	83	868736739512	20	10
211	84	868736741645	84	868736741625	20	10
212	85	868736743768	85	868736743738	30	15
213	86	868736745871	86	868736745861	10	5
214	87	868736747994	87	868736747974	20	10
215	88	868736750137	88	868736750087	50	25
216	89	868736752250	89	868736752210	40	20
217	90	868736754373	90	868736754323	50	25
218	91	868736756476	91	868736756446	30	15
219	92	868736758589	92	868736758559	30	15
220	93	868736760712	93	868736760672	40	20
221	94	868736762815	94	868736762795	20	10
222	95	868736764928	95	868736764908	20	10
223	96	868736767081	96	868736767021	60	30
224	97	868736769195	97	868736769144	51	25.5
225	98	868736771328	98	868736771257	71	35.5
226	99	868736773431	99	868736773371	60	30
227	100	868736775544	100	868736775494	50	25
228	101	868736777667	101	868736777607	60	30
229	102	868736779760	102	868736779720	40	20
230	103	868736781883	103	868736781843	40	20
231	104	868736783996	104	868736783956	40	20
232	105	868736786099	105	868736786079	20	10
233	106	868736788212	106	868736788192	20	10
234	107	868736790325	107	868736790305	20	10
235	108	868736792438	108	868736792428	10	5
236	109	868736794651	109	868736794541	110	55
237	110	868736796754	110	868736796654	100	50
238	111	868736798857	111	868736798777	80	40
239	112	868736800970	112	868736800890	80	40
240	113	868736803083	113	868736803013	70	35
241	114	868736805206	114	868736805126	80	40
242	115	868736807339	115	868736807239	100	50
243	116	868736809462	116	868736809362	100	50
244	117	868736811585	117	868736811475	110	55
245	118	868736813709	118	868736813598	111	55.5
246	119	868736815751	119	868736815721	30	15
247	120	868736817875	120	868736817834	41	20.5
248	121	868736819998	121	868736819958	40	20

249	122	868736822101	122	868736822071	30	15
250	123	868736824234	123	868736824184	50	25
251	124	868736826357	124	868736826307	50	25
252	125	868736828470	125	868736828420	50	25
253	0	868736830583	0	868736830533	50	25
254	1	868736832746	1	868736832656	90	45
255	2	868736834869	2	868736834769	100	50
256	3	868736836992	3	868736836892	100	50
257	4	868736839105	4	868736839005	100	50
258	5	868736841208	5	868736841128	80	40
259	6	868736843331	6	868736843241	90	45
260	7	868736845394	7	868736845354	40	20
261	8	868736847497	8	868736847477	20	10
262	9	868736849630	9	868736849590	40	20
263	10	868736851733	10	868736851713	20	10
264	11	868736853846	11	868736853826	20	10
265	12	868736855969	12	868736855939	30	15
266	13	868736858172	13	868736858072	100	50
267	14	868736860296	14	868736860185	111	55.5
268	15	868736862409	15	868736862298	111	55.5
269	16	868736864512	16	868736864421	91	45.5
270	17	868736866615	17	868736866534	81	40.5
271	18	868736868738	18	868736868658	80	40
272	19	868736870821	19	868736870771	50	25
273	20	868736872954	20	868736872884	70	35
274	21	868736875057	21	868736875007	50	25
275	22	868736877170	22	868736877120	50	25
276	23	868736879283	23	868736879233	50	25
277	24	868736881446	24	868736881356	90	45
278	25	868736883549	25	868736883469	80	40
279	26	868736885672	26	868736885582	90	45
280	27	868736887775	27	868736887705	70	35
281	28	868736889898	28	868736889818	80	40
282	29	868736892001	29	868736891941	60	30
283	30	868736894114	30	868736894054	60	30
284	31	868736896227	31	868736896167	60	30
285	32	868736898340	32	868736898290	50	25
286	33	868736900443	33	868736900403	40	20
287	34	868736902556	34	868736902516	40	20
288	35	868736904679	35	868736904639	40	20
289	36	868736906822	36	868736906752	70	35
290	37	868736908935	37	868736908865	70	35
291	38	868736911048	38	868736910988	60	30
292	39	868736913182	39	868736913101	81	40.5
293	40	868736915265	40	868736915214	51	25.5
294	41	868736917378	41	868736917338	40	20
295	42	868736919491	42	868736919451	40	20
296	43	868736921614	43	868736921574	40	20
297	44	868736923717	44	868736923687	30	15
298	45	868736925850	45	868736925810	40	20
299	46	868736927963	46	868736927923	40	20

300	47	868736930066	47	868736930046	20	10
301	48	868736932179	48	868736932159	20	10
302	49	868736934292	49	868736934272	20	10
303	50	868736936495	50	868736936385	110	55
304	51	868736938608	51	868736938508	100	50
305	52	868736940721	52	868736940621	100	50
306	53	868736942824	53	868736942744	80	40
307	54	868736944937	54	868736944857	80	40
308	55	868736947050	55	868736946970	80	40
309	56	868736949163	56	868736949093	70	35
310	57	868736951266	57	868736951206	60	30
311	58	868736953379	58	868736953329	50	25
312	59	868736955482	59	868736955442	40	20
313	60	868736957595	60	868736957565	30	15
314	61	868736959708	61	868736959678	30	15
315	62	868736961811	62	868736961791	20	10
316	63	868736964015	63	868736963915	100	50
317	64	868736966128	64	868736966028	100	50
318	65	868736968251	65	868736968151	100	50
319	66	868736970364	66	868736970264	100	50
320	67	868736972437	67	868736972387	50	25
321	68	868736974550	68	868736974500	50	25
322	69	868736976663	69	868736976613	50	25
323	70	868736978766	70	868736978736	30	15
324	71	868736980879	71	868736980849	30	15
325	72	868736983042	72	868736982962	80	40
326	73	868736985205	73	868736985085	120	60
327	74	868736987238	74	868736987198	40	20
328	75	868736989351	75	868736989321	30	15
329	76	868736991454	76	868736991434	20	10
330	77	868736993577	77	868736993547	30	15
331	78	868736995720	78	868736995670	50	25
332	79	868736997843	79	868736997783	60	30
333	80	868736999946	80	868736999896	50	25
334	81	868737002049	81	868737002019	30	15
335	82	868737004172	82	868737004132	40	20
336	83	868737006285	83	868737006245	40	20
337	84	868737008388	84	868737008368	20	10
338	85	868737010512	85	868737010481	31	15.5
339	86	868737012625	86	868737012605	20	10
340	87	868737014828	87	868737014718	110	55
341	88	868737016931	88	868737016831	100	50
342	89	868737019034	89	868737018954	80	40
343	90	868737021147	90	868737021067	80	40
344	91	868737023260	91	868737023190	70	35
345	92	868737025363	92	868737025303	60	30
346	93	868737027466	93	868737027416	50	25
347	94	868737029579	94	868737029539	40	20
348	95	868737031682	95	868737031652	30	15
349	96	868737033785	96	868737033775	10	5
350	97	868737035918	97	868737035888	30	15

351	98	868737038031	98	868737038001	30	15
352	99	868737040134	99	868737040124	10	5
353	100	868737042337	100	868737042237	100	50
354	101	868737044440	101	868737044360	80	40
355	102	868737046573	102	868737046473	100	50
356	103	868737048676	103	868737048596	80	40
357	104	868737050779	104	868737050719	60	30
358	105	868737052892	105	868737052832	60	30
359	106	868737055005	106	868737054945	60	30
360	107	868737057109	107	868737057068	41	20.5
361	108	868737059222	108	868737059181	41	20.5
362	109	868737061385	109	868737061295	90	45
363	110	868737063508	110	868737063418	90	45
364	111	868737065631	111	868737065531	100	50
365	112	868737067754	112	868737067654	100	50
366	113	868737069857	113	868737069767	90	45
367	114	868737071970	114	868737071880	90	45
368	115	868737074083	115	868737074003	80	40
369	116	868737076196	116	868737076116	80	40
370	117	868737078299	117	868737078239	60	30
371	118	868737080412	118	868737080352	60	30
372	119	868737082525	119	868737082465	60	30
373	120	868737084628	120	868737084578	50	25
374	121	868737086731	121	868737086701	30	15
375	122	868737088844	122	868737088814	30	15
376	123	868737090957	123	868737090937	20	10
377	124	868737093080	124	868737093050	30	15
378	125	868737095193	125	868737095173	20	10
379	0	868737097326	0	868737097286	40	20
380	1	868737099449	1	868737099399	50	25
381	2	868737101582	2	868737101522	60	30
382	3	868737103685	3	868737103635	50	25
383	4	868737105809	4	868737105748	61	30.5
384	5	868737107922	5	868737107872	50	25
385	6	868737110085	6	868737109985	100	50
386	7	868737112188	7	868737112098	90	45
387	8	868737114311	8	868737114221	90	45
388	9	868737116364	9	868737116334	30	15
389	10	868737118487	10	868737118457	30	15
390	11	868737120600	11	868737120570	30	15
391	12	868737122703	12	868737122683	20	10
392	13	868737124826	13	868737124806	20	10
393	14	868737126949	14	868737126919	30	15
394	15	868737129062	15	868737129042	20	10
395	16	868737131265	16	868737131155	110	55
396	17	868737133368	17	868737133268	100	50
397	18	868737135481	18	868737135391	90	45
398	19	868737137614	19	868737137504	110	55
399	20	868737139717	20	868737139617	100	50
400	21	868737141830	21	868737141740	90	45
401	22	868737143953	22	868737143853	100	50

402	23	868737146016	23	868737145976	40	20
403	24	868737148139	24	868737148089	50	25
404	25	868737150262	25	868737150202	60	30
405	26	868737152365	26	868737152325	40	20
406	27	868737154479	27	868737154438	41	20.5
407	28	868737156582	28	868737156552	30	15
408	29	868737158695	29	868737158665	30	15
409	30	868737160808	30	868737160788	20	10
410	31	868737162981	31	868737162901	80	40
411	32	868737165094	32	868737165024	70	35
412	33	868737167227	33	868737167137	90	45
413	34	868737169330	34	868737169250	80	40
414	35	868737171433	35	868737171373	60	30
415	36	868737173556	36	868737173486	70	35
416	37	868737175639	37	868737175609	30	15
417	38	868737177752	38	868737177722	30	15
418	39	868737179865	39	868737179835	30	15
419	40	868737181978	40	868737181958	20	10
420	41	868737184091	41	868737184081	10	5
421	42	868737186214	42	868737186194	20	10
422	43	868737188337	43	868737188307	30	15
423	44	868737190460	44	868737190430	30	15
424	45	868737192573	45	868737192543	30	15
425	46	868737194696	46	868737194666	30	15
426	47	868737196839	47	868737196779	60	30
427	48	868737199003	48	868737198892	111	55.5
428	49	868737201035	49	868737201015	20	10
429	50	868737203179	50	868737203128	51	25.5
430	51	868737205372	51	868737205252	120	60
431	52	868737207425	52	868737207365	60	30
432	53	868737209608	53	868737209488	120	60
433	54	868737211621	54	868737211601	20	10
434	55	868737213864	55	868737213714	150	75
435	56	868737215897	56	868737215837	60	30
436	57	868737218010	57	868737217950	60	30
437	58	868737220203	58	868737220063	140	70
438	59	868737222216	59	868737222186	30	15
439	60	868737224409	60	868737224299	110	55
440	61	868737226462	61	868737226422	40	20
441	62	868737228665	62	868737228535	130	65
442	63	868737230758	63	868737230648	110	55
443	64	868737232881	64	868737232771	110	55
444	65	868737234944	65	868737234884	60	30
445	66	868737237107	66	868737236997	110	55
446	67	868737239240	67	868737239120	120	60
447	68	868737241273	68	868737241233	40	20
448	69	868737243376	69	868737243346	30	15
449	70	868737245539	70	868737245469	70	35
450	71	868737247703	71	868737247582	121	60.5
451	72	868737249826	72	868737249695	131	65.5
452	73	868737251849	73	868737251818	31	15.5

453	74	868737253982	74	868737253932	50	25
454	75	868737256155	75	868737256045	110	55
455	76	868737258238	76	868737258168	70	35
456	77	868737260431	77	868737260281	150	75
457	78	868737262444	78	868737262404	40	20
458	79	868737264687	79	868737264517	170	85
459	80	868737266700	80	868737266640	60	30
460	81	868737268833	81	868737268753	80	40
461	82	868737270886	82	868737270866	20	10
462	83	868737273079	83	868737272979	100	50
463	84	868737275202	84	868737275102	100	50
464	85	868737277235	85	868737277215	20	10
465	86	868737279368	86	868737279338	30	15
466	87	868737281551	87	868737281451	100	50
467	88	868737283604	88	868737283564	40	20
468	89	868737285747	89	868737285687	60	30
469	90	868737287910	90	868737287800	110	55
470	91	868737289973	91	868737289913	60	30
471	92	868737292146	92	868737292036	110	55
472	93	868737294209	93	868737294149	60	30
473	94	868737296292	94	868737296272	20	10
474	95	868737298486	95	868737298385	101	50.5
475	96	868737300699	96	868737300498	201	100.5
476	97	868737302702	97	868737302622	80	40
477	98	868737304815	98	868737304735	80	40
478	99	868737306918	99	868737306848	70	35
479	100	868737309061	100	868737308971	90	45
480	101	868737311134	101	868737311084	50	25
481	102	868737313297	102	868737313197	100	50
482	103	868737315390	103	868737315320	70	35
483	104	868737317553	104	868737317433	120	60
484	105	868737319596	105	868737319546	50	25
485	106	868737321769	106	868737321669	100	50
486	107	868737323842	107	868737323782	60	30
487	108	868737325985	108	868737325895	90	45
488	109	868737328128	109	868737328018	110	55
489	110	868737330271	110	868737330131	140	70
490	111	868737332344	111	868737332244	100	50
491	112	868737334397	112	868737334367	30	15
492	113	868737336590	113	868737336480	110	55
493	114	868737338723	114	868737338593	130	65
494	115	868737340836	115	868737340716	120	60
495	116	868737342950	116	868737342829	121	60.5
496	117	868737345012	117	868737344942	70	35
497	118	868737347116	118	868737347065	51	25.5
498	119	868737349299	119	868737349178	121	60.5
499	120	868737351422	120	868737351292	130	65
500	121	868737353485	121	868737353415	70	35
501	122	868737355548	122	868737355528	20	10
502	123	868737357761	123	868737357641	120	60
503	124	868737359844	124	868737359764	80	40

504	125	868737361907	125	868737361877	30	15
505	0	868737364030	0	868737364000	30	15
506	1	868737366213	1	868737366113	100	50
507	2	868737368246	2	868737368226	20	10
508	3	868737370449	3	868737370349	100	50
509	4	868737372532	4	868737372462	70	35
510	5	868737374685	5	868737374585	100	50
511	6	868737376818	6	868737376698	120	60
512	7	868737378931	7	868737378811	120	60
513	8	868737380984	8	868737380934	50	25
514	9	868737383157	9	868737383047	110	55
515	10	868737385190	10	868737385160	30	15
516	11	868737387303	11	868737387283	20	10
517	12	868737389476	12	868737389396	80	40
518	13	868737391640	13	868737391519	121	60.5
519	14	868737393672	14	868737393632	40	20
520	15	868737395786	15	868737395745	41	20.5
521	16	868737397959	16	868737397868	91	45.5
522	17	868737400102	17	868737399982	120	60
523	18	868737402245	18	868737402095	150	75
524	19	868737404278	19	868737404218	60	30
525	20	868737406351	20	868737406331	20	10
526	21	868737408504	21	868737408444	60	30
527	22	868737410687	22	868737410567	120	60
528	23	868737412700	23	868737412680	20	10
529	24	868737414873	24	868737414803	70	35
530	25	868737417046	25	868737416916	130	65
531	26	868737419119	26	868737419029	90	45
532	27	868737421272	27	868737421142	130	65
533	28	868737423315	28	868737423265	50	25
534	29	868737425498	29	868737425378	120	60
535	30	868737427561	30	868737427501	60	30

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Suite 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   Monterey, CA 93943

3. Center for Naval Analysis 1
   4401 Ford Ave.
   Alexandria, VA 22302

4. Dr. Ted Lewis, Chairman, Code CS/L 1
   Computer Science Dept.
   Naval Postgraduate School
   Monterey, CA 93943

5. Chief of Naval Research 1
   800 North Quincy St.
   Arlington, VA 22217

6. Dr. Luqi, Code CS/Lq 1
   Computer Science Dept.
   Naval Postgraduate School
   Monterey, CA 93943

7. Dr. Marvin Langston 1
   1225 Jefferson Davis Highway
   Crystal Gateway 2 / Suite 1500
   Arlington ,VA 22202-4311

8. David Hislop 1
   U.S. Army Research Office
   PO Box 12211
   Research Triangle Park, NC 27709-2211

9. Capt. Talbot Manvel 1
   Naval Sea System Command
   2531 Jefferson Davis Hwy.
   Attn: TMS 378 Capt. Manvel
   Arlington, VA 22240-5150

10. CDR Michael McMahon 1
Naval Sea System Command
2531 Jefferson Davis Hwy.
Arlington, VA 22242-5160

11. Dr. Elizabeth Wald 1
Office Of Naval Research
800 N. Quincy St.
ONR CODE 311
Arlington , VA 22217-5660

12. Dr. Ralph Wachter 1
Office of Naval Research
800 N. Quincy St.
CODE 311
Arlington, VA 22217-5660

13. Army Research Lab 1
115 O'Keefe Building
Attn: Mark Kendall
Atlanta, GA 30332-0862

14. National Science Foundation 1
Attn: Bruce Barnes
Div. Computer & Computation Research
1800 G St. NW
Washington, DC 20550

15. National Science Foundation 1
Attn: Bill Agresty
4201 Wilson Blvd.
Arlington, VA 22230

16. Hon. John W. Douglas 1
Assistant Secretary of the Navy
(Research, Devlopment and Aquisition)
Room E741
1000 Navy Pentagon
Washingotn, DC 20350-1000

17. VADM Herbert A. Brown, Commander Third Fleet 1
2421 Vella La Vella Rd.
NAB Coronado
San Diego, CA 92155-5490

18. RADM George F. A. Wagner, Code 00 1
SPAWAR
53560 Oceanview Dr. Ste.317
San Diego, CA 92152-5088

19. Paul Wessel, Code 07 1
SPAWAR
53560 Oceanview Dr. Ste.317
San Diego, CA 92152-5088

20. Capt. Harold Williams, Code D00 1
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

21. Dr. Robert Kolb, Code D01 1
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

22. Rod Smith, Code D40 1
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

23. Jim Price, Code D43 1
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

24. Rick Millen, Code D431 1
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

25. Rich Heidecker, Code D432 1
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

26. Floyd Bailey, Code D4322 4
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

27. Carl Robbins, Code D4323 4
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001

28. Technical Library Branch, Code D0274 1
NCCOSC RDT&E Division
53560 Hull Street
San Diego, CA 92152-5001